Trabajo Fin de Máster
Máster en Ingeniería Industrial

Adaptación física y electrónica de un actuador eléctrico a un robot manipulador

Autor: José Luis Pozo Acosta
Tutores: David Muñoz de la Peña Sequedo
Luis Fernando Castaño Castaño

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Máster
Máster en Ingeniería Industrial

Adaptación física y electrónica de un actuador eléctrico a un robot manipulador

Autor:
José Luis Pozo Acosta

Tutores:
David Muñoz de la Peña Sequedo
Luis Fernando Castaño Castaño

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Agradecimientos

A mi familia, por haberme apoyado en todas las etapas de mi vida.

A mis amigos, por haber elegido seguir sufriendo juntos en esta escuela.

A mis tutores David Muñoz y Fernando Castaño, por ofrecerme su ayuda y la oportunidad de llevar a cabo este proyecto.

A Alicia, por hacer conmigo este trabajo sin estar matriculada en este Máster.

José Luis Pozo Acosta

Sevilla, 2018

La teoría es cuando lo sabes todo, pero nada funciona.
La práctica es cuando todo funciona, pero nadie sabe porqué.
En nuestro laboratorio la teoría y la práctica se combinan, nada funciona y nadie sabe porqué.

Resumen

La idea de este proyecto surgió tras la adquisición de un nuevo modelo de robot industrial *IRB120* de *ABB* para el laboratorio de robótica. Dicho manipulador requiere de un elemento terminal que le permita realizar una tarea específica. En nuestro caso, se ha optado por un elemento de sujeción, o pinza, en particular el modelo *LEHZ25K2-14* de la empresa *SMC Corporation*.

La pinza pertenece a una gama de actuadores eléctricos que son controlados por un driver programable de la serie *LEC-P6*, el cual se programa a través del software *ACT Controller*.

Una vez configurado el modo de funcionamiento del driver, éste ha de ser conectado a la controladora del robot, *IRC5*, cableando sus entradas y salidas digitales para permitir la comunicación entre dichos sistemas.

El problema aparece cuando este enlace requiere de un total de 24 conexiones entre entradas y salidas, lo cual limita la capacidad del robot para ser interconectado con otros sistemas periféricos. Ello se soluciona interponiendo un tercer actor entre la controladora del robot y la de la pinza. La solución adoptada pasa por usar un *Arduino*, el cual será el encargado de gestionar y procesar la comunicación entre ambos elementos.

Para adaptar los niveles de tensión entre *Arduino*, el cual funciona a 5V, y los controladores del robot y la pinza, los cuales trabajan a 24V, se ha desarrollado una PCB en la que se hace uso de optoacopladores para suplir los niveles de tensión necesarios a los sistemas. Dicha PCB también hace las veces de HMI, permitiendo al usuario obtener información del proceso.

Además, haciendo uso del software de modelado CAD *CatiaV5*, y de las nuevas tecnologías de impresión 3D, se han desarrollado y fabricado diversas piezas necesarias para la interconexión física de los componentes y el correcto funcionamiento de los mismos.

Por último, se han diseñado los diversos elementos necesarios para recrear en su totalidad una estación de trabajo que reproduzca las condiciones geométricas exactas del laboratorio de robótica de la Escuela Técnica Superior de Ingeniería de Sevilla. Dicha estación ha sido modelada con el software de programación de robots industriales *RobotStudio* y el lenguaje *RAPID*, ambos desarrollados por *ABB*.

Abstract

The idea for this project came after the acquisition of a new industrial robot model IRB120 from ABB for the robotics laboratory. Such a manipulator requires a terminal element that allows it to perform a specific task. In our case, we have opted for a fastening element, or clamp, in particular the model LEHZ25K2-14 of the company SMC Corporation.

The clamp belongs to a range of electric actuators that are controlled by a programmable driver of the LEC-P6 series, which is programmed through the ACT Controller software.

Once the operating mode of the driver is configured, it must be connected to the controller of the robot, IRC5, wiring its digital inputs and outputs to allow communication between these systems.

The problem appears when this link requires a total of 24 connections between inputs and outputs, which limits the robot's ability to be interconnected with other peripheral systems. This is solved by interposing a third actor between the controller of the robot and the one of the clamp. The solution adopted is to use an Arduino, which will be in charge of managing and processing the communication between both elements.

To adapt the voltage levels between Arduino, which works at 5V, and the controllers of the robot and the clamp, which work at 24V, a PCB has been developed in which optocouplers are used to supply the necessary voltage levels to systems. This PCB also acts as HMI, allowing the user to obtain information about the process.

In addition, using CatiaV5 CAD modeling software and new 3D printing technologies, various pieces have been developed and manufactured necessary for the physical interconnection of the components and their correct operation.

Finally, we have designed the various elements necessary to fully recreate a workstation that reproduces the exact geometric conditions of the robotics laboratory of the Technical School of Engineering of Seville. This station has been modeled with the RobotStudio industrial robot programming software and the RAPID language, both developed by ABB.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN Y OBJETIVOS

El laboratorio de robótica, ubicado en la Escuela Técnica Superior de Ingeniería de Sevilla, cuenta actualmente con equipos de robots manipuladores para instruir a los alumnos acerca de la programación y el funcionamiento de la robótica industrial. Los más utilizados por los alumnos que se inician en el campo de la robótica son los modelos *SCORBOT-ER V*.

Dicho sistema consiste en un robot articulado de 5 grados de libertad que se complementa con una pinza, una controladora y una pistola de programación.

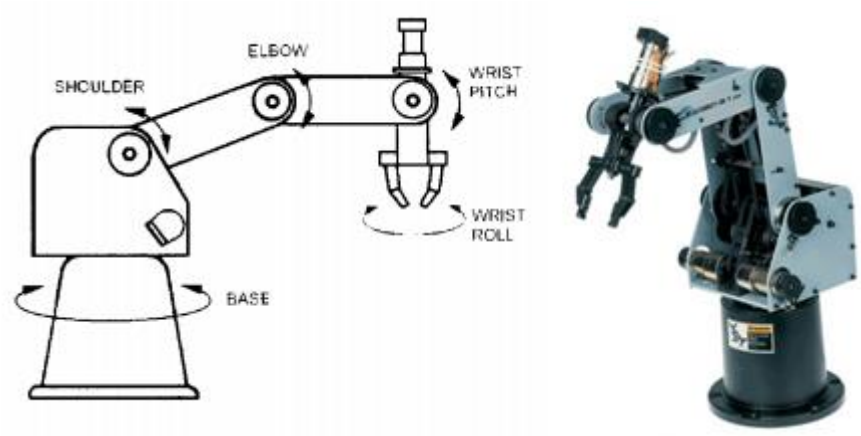


Figura 1: Scorbot-ER V

Aunque el manipulador tiene las suficientes características técnicas como para realizar tareas básicas que permiten desarrollar aplicaciones simples, su modo de programación queda obsoleto en comparación a lo que podemos llegar a encontrarnos hoy en día en la industrial real, así como propiedades de resolución, exactitud y repetibilidad.

Es por ello que el Departamento de Ingeniería de Sistemas y Automática decidió invertir en la adquisición de un nuevo modelo de sistema robótico acorde con el mercado actual. El producto elegido es el *IRB120* de la empresa *ABB*.

El sistema incluye un robot manipulador con una capacidad útil de 3 Kg y un alcance de 0.58 m, la controladora del robot denominada *IRC5* y una pistola de programación interactiva *Flexpendant*.

Figura 2: Sistema Robot IRB120 - Controladora IRC – Flexpendant

Como bien es sabido, un robot manipulador carece de utilidad si no se le dota de un elemento terminal que le permita realizar alguna tarea específica. Es por ello que se decide adquirir un actuador eléctrico de la marca *SMC*, en concreto una pinza de dos dedos serie *LEHZ* junto con su controlador, un driver programable para motor paso a paso *LEC-P6*.

Figura 3: Pinza LEHZ – Controlador LECP6

Debido a la gran cantidad de señales digitales de entrada y salida que son necesarias para el control de la pinza, la conexión directa entre el bornero E/S del controlador del robot y el controlador *LEC-P6*, dejaría muy reducido el número de señales disponibles por el robot para el control de otros actuadores y/o sensores externos.

La solución adoptada consiste en integrar un *Arduino Mega 2560* en una PCB (*Printed Circuit Board* o Placa de Circuito Impreso) para que lleve a cabo la gestión de las señales de control necesarias entre el robot y la pinza.

Figura 4: Arduino Mega 2560 – Esquemático de la PCB implementada

Además de la adaptación electrónica, fue necesaria una interconexión física entre la muñeca del robot y el actuador eléctrico. Para ello, contando con los planos de ambos sistemas, se diseñó, a través de *CatiaV5*, una serie de piezas que, una vez fabricadas con tecnologías de fabricación aditiva, permiten la conexión y el buen funcionamiento del sistema.

Figura 5: Algunas piezas impresas en 3D

Por último, haciendo uso del software *RobotStudio* de *ABB*, se ha integrado la pinza digitalmente como una herramienta más de la librería de componentes. Del mismo modo, se ha diseñado una estación de trabajo que permite al usuario trabajar en un entorno simulado de condiciones físicas exactas al laboratorio de Robótica de la Escuela de Ingeniería.

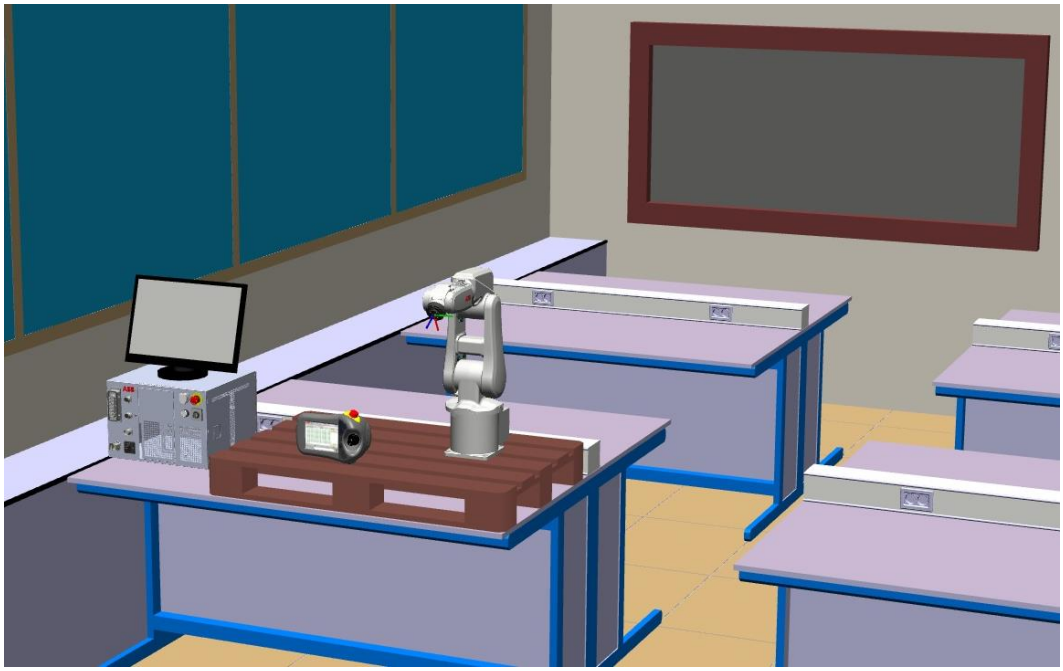


Figura 6: Laboratorio de robótica en RobotStudio

2 DESCRIPCIÓN DE LOS COMPONENTES

2.1 Introducción

A continuación se exponen las características técnicas de los componentes utilizados para la realización de este trabajo, así como los elementos que integran todo el sistema.

Para obtener información más detalla sobre datos técnicos o conexiones entre sistemas, acudir a los siguientes anexos:

- Anexo A: *Esquema general de conexión de los componentes*
- Anexo B: *Diagramas de conexión del controlador LEC-P6*
- Anexo C: *Diagrama de conexión de la PCB*
- Anexo D: *Diagrama de conexión del controlador IRC5*
- Anexo E: *Variables*
- Anexo G: *Manuales de Operación, Productos y Datasheets*

➢ **Nota:** Hacer click sobre los enlaces azules para acudir a los documentos.

2.2 ABB

2.2.1 Robot IRB120

El *IRB120* es el robot industrial de 6 ejes más reciente de *ABB Robotics*. Cuenta con una carga útil de 3 kg y está diseñado específicamente para industrias de fabricación que utilizan una automatización flexible basada en robot.
Tiene una estructura abierta especialmente adaptada para un uso flexible y presenta unas grandes posibilidades de comunicación con sistemas externos.

El robot está equipado con el controlador *IRC5 Compact* o *IRC5* (*Single Cabinet Controller*) y el software de control de robots *RobotWare*, el cual admite todos los aspectos del sistema de robot, como el control del movimiento, el desarrollo y la ejecución de programas, la comunicación, etc

Para disponer de una funcionalidad mayor, es posible equipar al robot con software opcionales, para compatibilizar con determinadas aplicaciones, como la aplicación de adhesivo y la soldadura, funciones de comunicación o comunicaciones de red, además de funciones avanzadas como el procesamiento multitarea, el control de sensores, etc.

Peso del robot	25 Kg
Capacidad de manejo	3 Kg
Alcance	0.58 m
Nivel de ruido	<78 dB

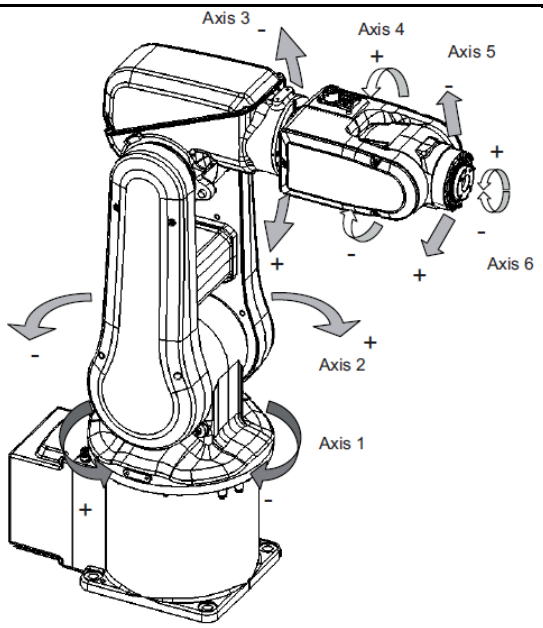
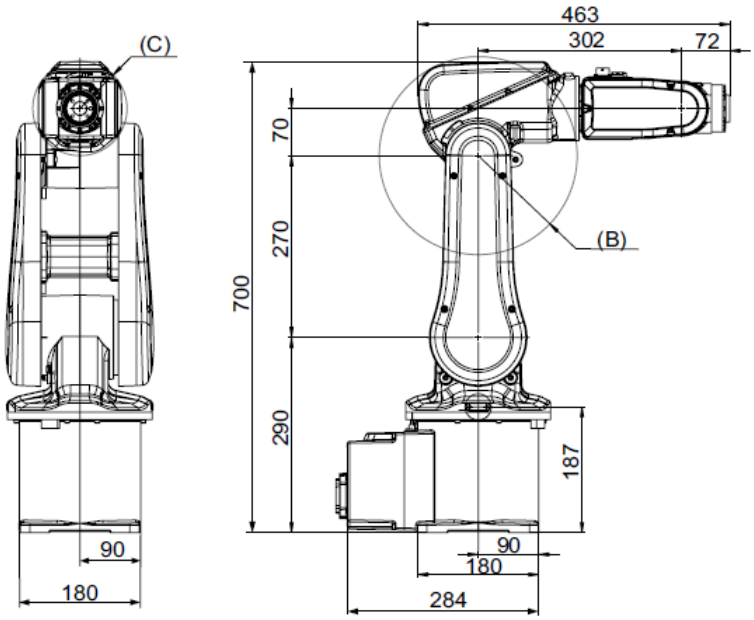
Ejes del manipulador	
Dimensiones	

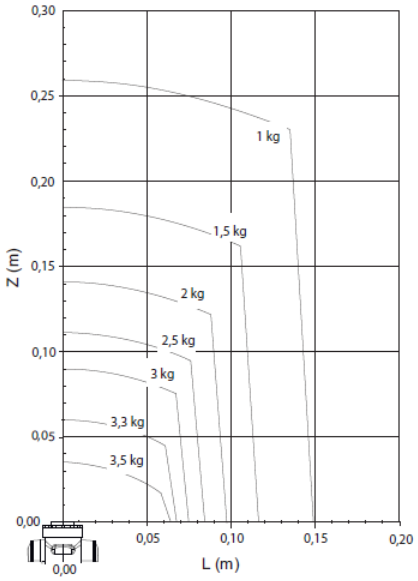
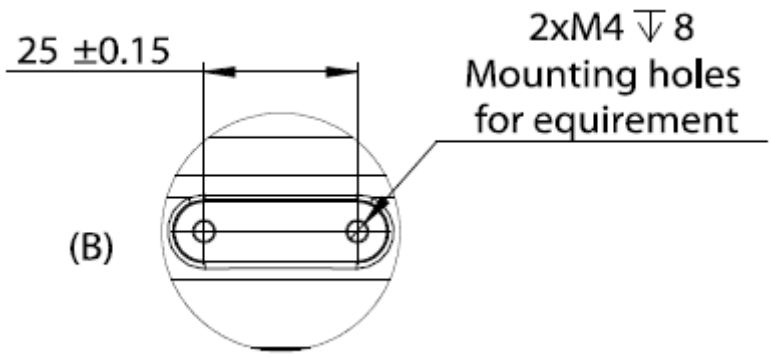
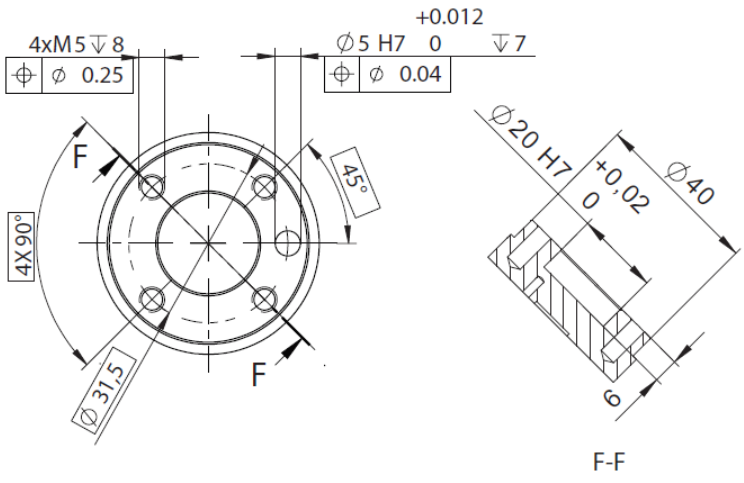
Diagrama de carga máxima	
Orificios para montajes de equipos adicionales	
Brida para herramientas del robot	

Tabla 1: Características técnicas y físicas del IRB120

2.2.2 Controlador IRC5-C

El controlador *IRC5* contiene los elementos electrónicos necesarios para gestionar el manipulador, los ejes adicionales y equipos periféricos. En nuestro caso, contamos con un controlador compacto *IRC5-C*.

El controlador *IRC5 Compact* es un controlador de robot de escritorio diseñado para ocupar un menor espacio, limitando algunas características físicas de otros controladores. Por ejemplo, en el modelo Compact sólo puede montarse una unidad de E/S (se incluye *DSQC652* de serie) dentro del armario.

Peso del armario	28,5 Kg
Potencia consumida	250 W

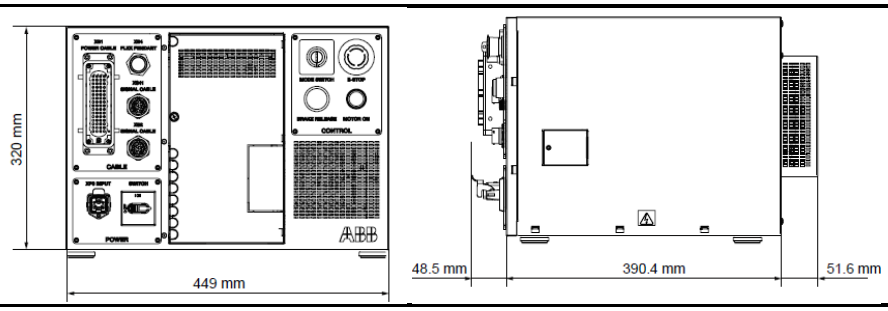
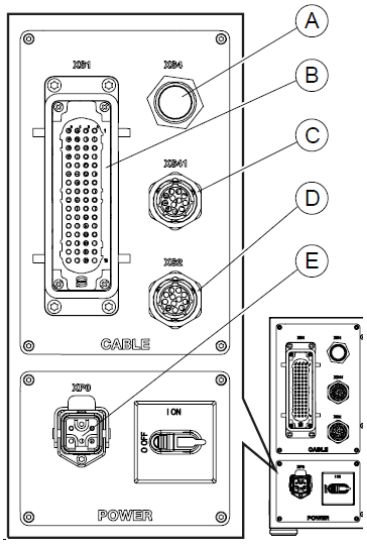
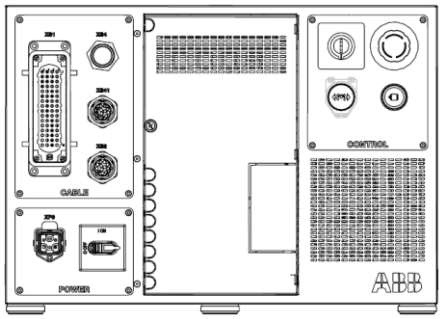
Medidas	
Interfaz de conexiones	 **A** Conexión de *Flexpendant* **B** Alimentación del robot **C** Tarjeta de medida serie **D** Tarjeta de medida serie del robot **E** Conexión eléctrica principal
Botonera del panel frontal	 **A** Interruptor de alimentación **B** Liberación de frenos **C** Selector de modo **D** Motores ON **E** Paro de emergencia

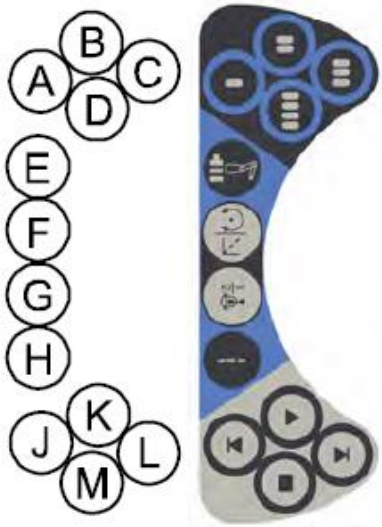
Tabla 2: Características técnicas y físicas del IRC5-C

2.2.3 Flexpendant

El *FlexPendant* es una unidad de operador de mano que se usa para realizar muchas de las tareas implicadas en el manejo de un sistema de robot: ejecutar programas, mover el manipulador, modificar programas del robot, etc. El *FlexPendant* ha sido diseñado para un funcionamiento continuo en entornos industriales agresivos.

El *FlexPendant* se compone tanto de hardware como de software y es un ordenador completo por sí solo. Forma parte del IRC5 y se conecta al controlador mediante un cable y un conector integrados. Sin embargo, la opción de pulsador de hot plug hace posible la desconexión del *FlexPendant* en el modo automático y seguir trabajando sin él.

<table>
<tr>
<td>Componentes Principales</td>
<td>


<table>
<tr><td>A</td><td>Conector</td></tr>
<tr><td>B</td><td>Pantalla táctil</td></tr>
<tr><td>C</td><td>Paro de emergencia</td></tr>
<tr><td>D</td><td>Joystick</td></tr>
<tr><td>E</td><td>Puerto USB</td></tr>
<tr><td>F</td><td>Habilitación</td></tr>
<tr><td>G</td><td>Puntero</td></tr>
<tr><td>H</td><td>Reset</td></tr>
</table>
</td>
</tr>
<tr>
<td>Botones de Hardware</td>
<td>


<table>
<tr><td>A-D</td><td>Teclas programables</td></tr>
<tr><td>E</td><td>Selección de unidad mecánica</td></tr>
<tr><td>F</td><td>Modos: Lineal - Reorientación</td></tr>
<tr><td>G</td><td>Modos: ejes 1-3 / ejes 4-6</td></tr>
<tr><td>H</td><td>Activar/desactivar incrementos</td></tr>
<tr><td>J</td><td>RETROCEDER</td></tr>
<tr><td>K</td><td>INICIAR</td></tr>
<tr><td>L</td><td>AVANZAR</td></tr>
<tr><td>M</td><td>DETENER</td></tr>
</table>
</td>
</tr>
</table>

Tabla 3: Características técnicas y físicas del Flexpendant

2.3 SMC

2.3.1 Pinza LEHZ25K2-14

La pinza *LEHZ25K2-14* es un actuador eléctrico controlado por un driver programable para motor paso a paso a 24V DC. Cuenta con un encoder incremental de 800 pulsos por vuelta. Para programar el mecanismo, solo es necesario introducir los datos a través de un software de fácil manejo para PC.

Carrera de apertura/cierre	14 mm
Fuerza de amarre	11 a 28 N
Velocidad de apertura/cierre	5 a 100 mm/s
Repetitividad	0.02 mm
Temperatura de trabajo	5 a 40 ºC
Peso	520 g
Tipo de motor	Motor paso a paso (Servo 24V DC)
Encoder	Fase A/B incremental (800 pulsos/giro)

1	Cuerpo
2	Placa del motor
3	Anillo guía
4	Tuerca trapecial
5	Husillo trapecial
6	Rodillo del tornillo
7	Rodillo del tornillo
8	Placa del cuerpo
9	Cubierta antipolvo
10	Conjunto de dedos
11	Cubierta del encoder
12	Palanca
13	Motor paso a paso

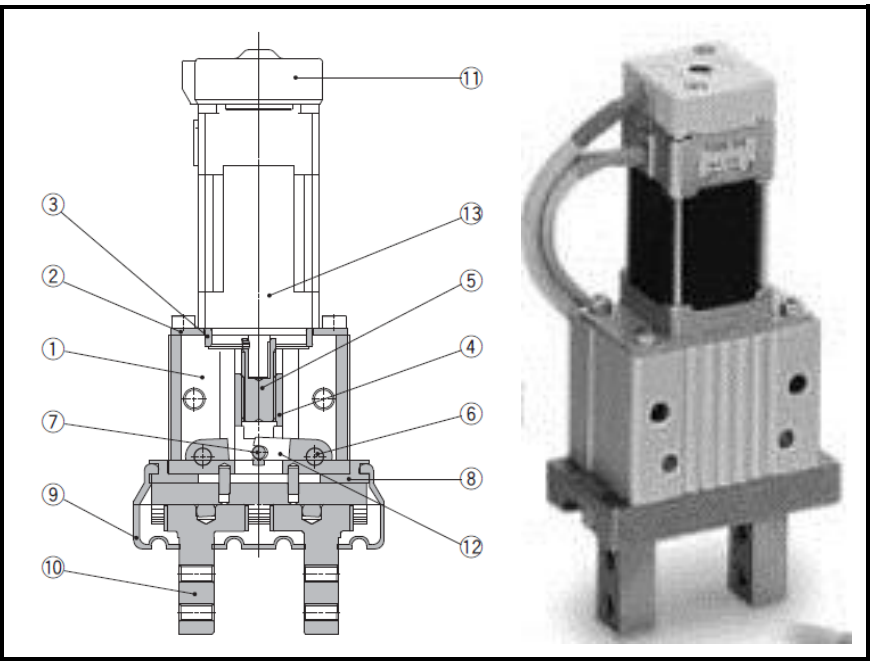


Tabla 4: Características técnicas y físicas de la pinza LEHZ25K2-14

2.3.2 Controlador LEC-P6

El controlador *LEC-P6* es el driver que controla el motor paso a paso de la pinza. Recibe alimentación de 24V. Puede programarse mediante software en PC, y su operación se controla mediante un dispositivo externo (PLC u otros) a través de 11 entradas y 13 salidas digitales a 24V.

Motor compatible	Motor paso a paso 24V DC
Tensión de alimentación	24V DC ± 10%
E/S	11 entradas / 13 salidas
Nº de puntos de posicionamiento	64
Comunicación en serie	RS485 Modbus
Memoria	EEPROM
Peso	170 g

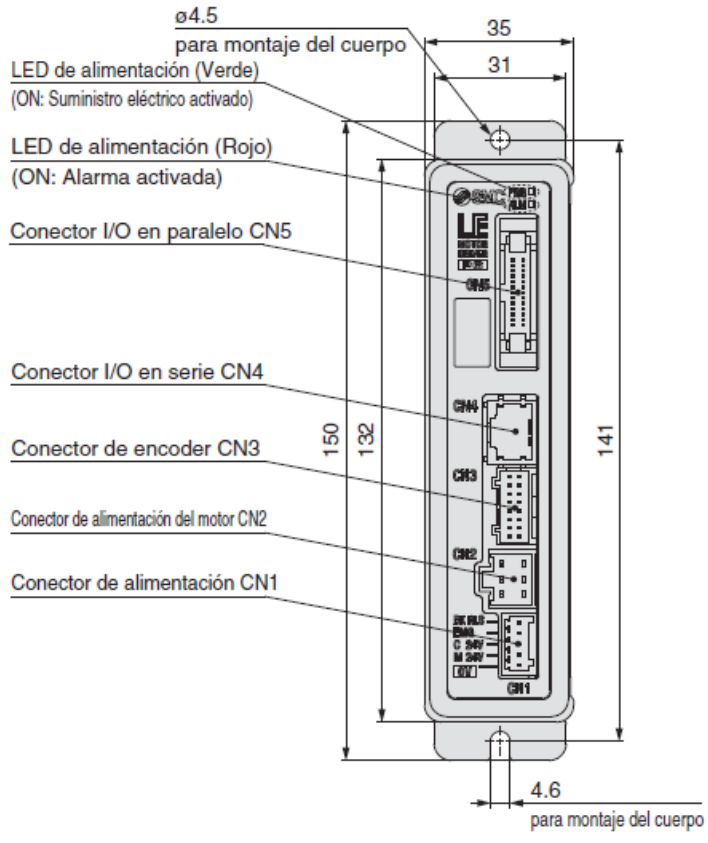
Dimensiones y Puertos	

Tabla 5: Características técnicas del controlador LEC-P6

2.4 Arduino

Arduino Mega es una plataforma de prototipos electrónica de código abierto basada en hardware y software flexibles y fáciles de usar.

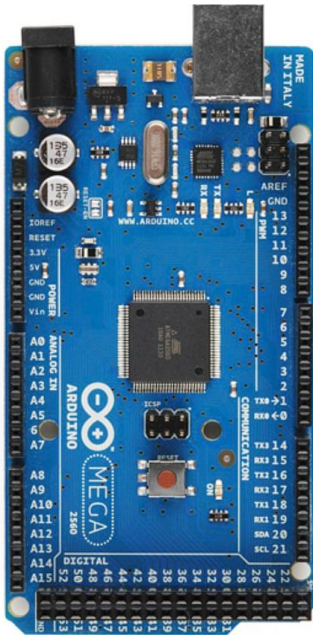
Microcontrolador	ATMega2560	
Voltaje de operación	5V	
Voltaje de entrada	7-12V	
Pines E/S Digitales	54	
Pines E Analógicos	16	
Pines S PWM	14	
Corriente por pines E/S	40 mA	
Memoria Flash	256 KB	
EEPROM	4 KB	
Reloj	16 MHz	

Tabla 6: Características técnicas de Arduino Mega

2.5 Optoacopladores

El *ISQ201* y el *CNY74-4H* son dos optoacopladores con un LED de infrarrojos GaAlAs y un fototransistor NPN de silicio. La información de la señal, incluyendo un nivel de corriente continua, se puede transmitir por el dispositivo, manteniendo el aislamiento galvánico entre la entrada y la salida. Este tipo de aislamiento se usa cuando se desea que se transmitan señales entre las distintas partes funcionales, pero las masas tienen que mantenerse separadas. Este aislamiento entre las masas o tierra se hace por motivos de seguridad.

2.5.1 ISQ201

ABSOLUTE MAXIMUM RATINGS
(25°C unless otherwise specified)

Storage Temperature	-40°C to +125°C
Operating Temperature	-25°C to +100°C
Lead Soldering Temperature (1/16 inch (1.6mm) from case for 10 secs)	260°C

INPUT DIODE

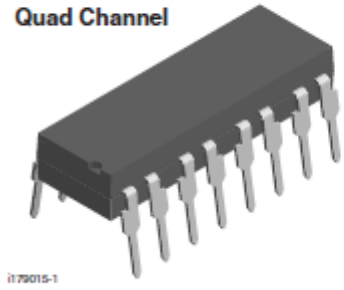
Forward Current	50mA
Reverse Voltage	6V
Power Dissipation	70mW

OUTPUT TRANSISTOR

Collector-emitter Voltage BV_{CEO}	70V
Emitter-collector Voltage BV_{ECO}	6V
Collector Current	50mA
Power Dissipation	150mW

POWER DISSIPATION

Total Power Dissipation (derate linearly 2.67mW/°C above 25°C)	170mW



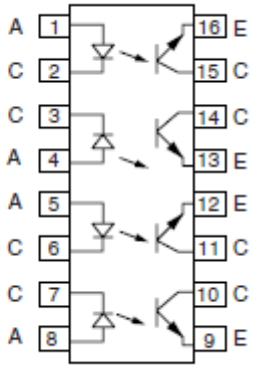


Tabla 7: Características técnicas del optoacoplador ISQ201

2.5.2 CNY74-4H

ABSOLUTE MAXIMUM RATINGS					
PARAMETER	TEST CONDITION	PART	SYMBOL	VALUE	UNIT
INPUT					
Peak reverse voltage			V_R	3	V
Forward continuous current			I_F	60	mA
Power dissipation			P_{diss}	100	mW
Derate linearly from 55 %				1.33	mW/°C

ABSOLUTE MAXIMUM RATINGS					
PARAMETER	TEST CONDITION	PART	SYMBOL	VALUE	UNIT
OUTPUT					
Collector emitter breakdown voltage			BV_{CEO}	70	V
Emitter collector breakdown voltage			BV_{ECO}	7	V
Power dissipation			P_{diss}	150	mW
Derate linearly from 25 °C				2	mW/°C

Tabla 8: Características técnicas del optoacoplador CNY74-4H

2.6 Fuentes de alimentación e interruptor

2.6.1 Fuente 24V

Voltaje nominal de entrada	110 – 240 VAC	
Voltaje nominal de salida	24 VDC	
Frecuencia	47 – 63 Hz	
Corriente de entrada	0.7 A	
Corriente de salida	2.5 A	
Protección	Fusible 2.5 A	

2.6.2 Fuente 9V

Voltaje nominal de entrada	110 – 240 VAC	
Voltaje nominal de salida	9 VAC	
Frecuencia	50 Hz	
Corriente de entrada	0.7 A	
Corriente de salida	1.8 A	
Conector	Jack 5.5 x 2 mm	

2.6.3 Interruptor bipolar

Tensión máxima	300V	
Intensidad máxima	16 A	
Nº de circuitos	2	
Indicador	Led rojo	

3 DESARROLLO DE LA PCB

3.1 Introducción

La fabricación de la PCB responde a dos necesidades. La primera de ellas consiste en resolver la incompatibilidad de los niveles de tensión entre *Arduino Mega*, el controlador *LEC-P6* y la controladora del robot *IRC5*. La segunda razón para desarrollar esta placa es la de integrar de una forma compacta y eficiente todas las conexiones entre los distintos sistemas, así como servir de base para elementos como indicadores leds, potenciómetro o interruptores.

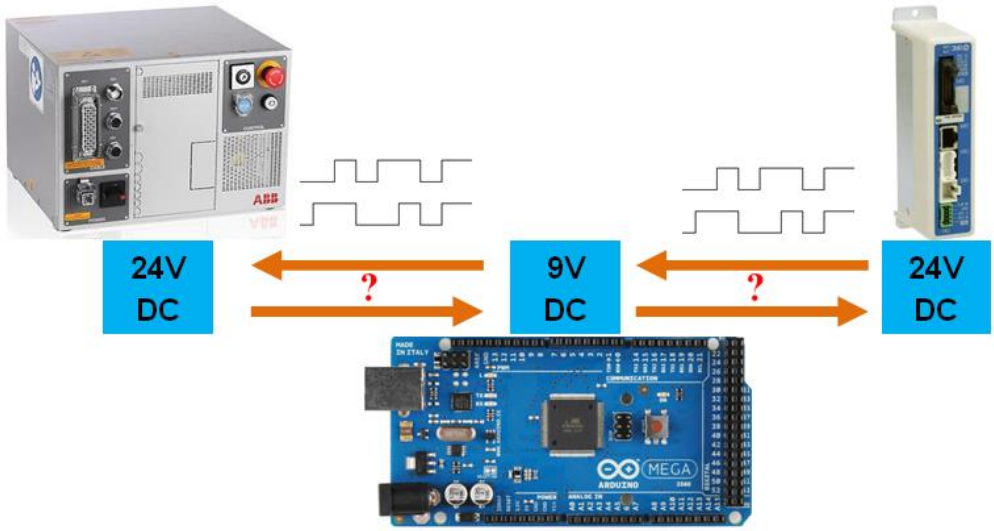


Figura 7: Incompatibilidad de tensiones

3.2 Optoacopladores

La solución adoptada para resolver estos saltos de tensión ha sido utilizar optoacopladores. Un optoacoplador es un dispositivo de emisión y recepción que funciona como un interruptor activado mediante la luz emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor. De este modo se combinan en un solo dispositivo semiconductor, un fotoemisor y un fotorreceptor cuya conexión entre ambos es óptica. Estos elementos se encuentran dentro de un encapsulado de tipo DIP.

Concretamente, los dispositivos utilizados son los integrados *ISQ-201* y *CNY74-4H*. Ambos cuentan con emcapsulado DIP de 16 pines y cada uno de ellos dispone internamente de 4 optoacopladores.

Las conversiones que se deben hacer son desde 24V (*IRC5* y *LEC-P6*) a 5V (*Arduino Mega*) y viceversa. Para ello, en ambos casos se han realizado montajes con resistencias de *pull down* para que el nivel de tensión a la salida sea siempre 0V o 24V / 5V. Además, a la entrada del diodo hay que incluir una resistencia que limite la corriente de entrada a un nivel suficiente para polarizar la base del transistor.

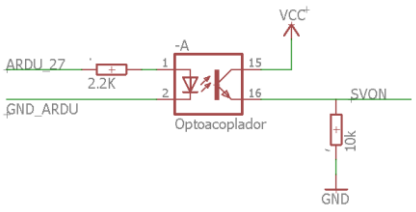
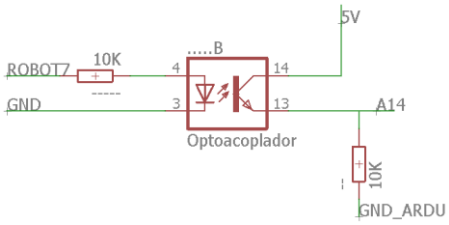
Entrada	0-5V (*Arduino*)	0-24V (*LEC-P6* / *IRC5*)
Salida	0-24V (*LEC-P6* / *IRC5*)	0-5V (*Arduino*)
Resistencia de entrada	2.2 KΩ	10 KΩ
Resistencia pull down	10 KΩ	10 KΩ
GND del diodo	GND de *Arduino*	GND de la fuente 24V
GND del fototransistor	GND de la fuente 24V	GND de *Arduino*
Intensidad de entrada	40 mA (*Arduino*)	10 mA (LEC-P6)
		

Tabla 9: Datos del montaje optoacoplado

La solución adoptada, además de resolver el problema de la compatibilidad, garantiza el aislamiento galvánico entre los sistemas, evitando que las corrientes fluyan de un equipo a otro, y manteniendo las diversas tierras separadas.

3.3 Eagle PCB

EAGLE, (*Easily Applicable Graphical Layout Editor*) es un programa de diseño de circuitos impresos y PCBs. El programa trae incluidas bibliotecas de componentes, sencillas y fáciles de manejar. Su entorno grafico ayuda al usuario a rutar los circuitos que interconectan los diversos componentes.

El software se divide en dos partes, *Schematic* y *Board*. En la pestaña del esquemático, se insertan los componentes del sistema y se interconectan nombrando los puertos con las etiquetas de las variables utilizadas.

En el Anexo E podemos encontrar más información sobre las variables del sistema.

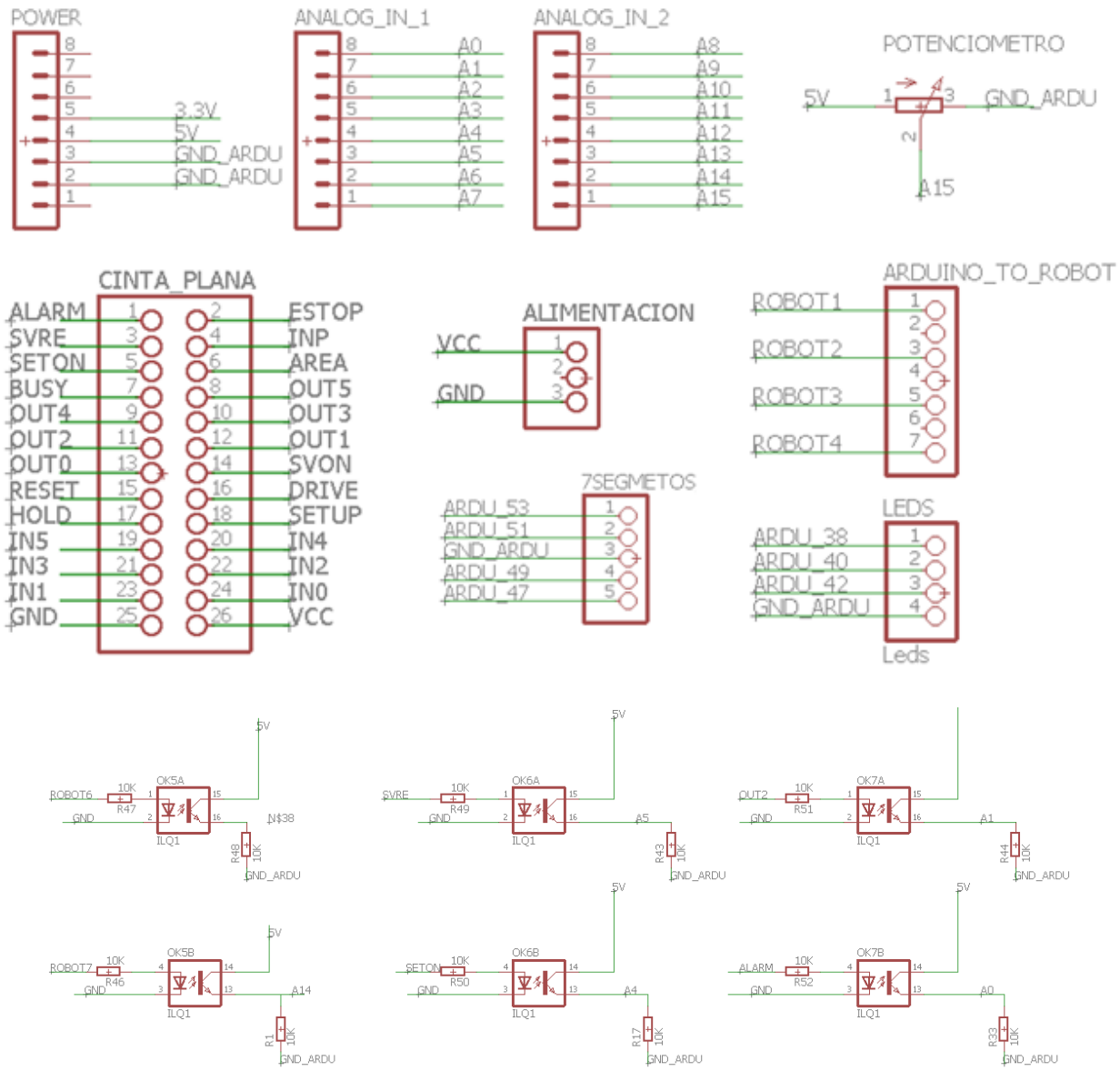


Figura 8: Algunos componentes del esquemático de la PCB

Una vez montado el circuito, el siguiente paso es rutarlo. En la segunda parte del programa, pestaña *Board*, hay que conectar los componentes debidamente entre ellos. Cada línea corresponde a una pista de cobre. A la cara superior de la placa se le llama *TOP*, la cual es la cara en la que se insertan los componentes. La cara inferior se denomina *BOTTOM*. Para pasar de una cara a otra, se coloca una *vía*, que no es más que un taladro en la placa para que pueda ser soldado un cable que interconectará ambas pistas.

En la imagen, podemos apreciar que han sido necesarios 8 encapsulados DIP (optoacopladores) para poder realizar todas las conversiones. Empezando por la esquina superior izquierda, tenemos los pines que deberán ser conectados a la fuente de alimentación de 24V y a su respectiva tierra (*24V+* y *GND*). Por debajo, tenemos las salidas a 24V hacia la controladora del robot (*PINZA -> ROBOT*). A continuación, se sitúa el conector de cinta plana / *CN5* que va a la controladora de la pinza *LEC-P6*. Le siguen las entradas provenientes de la controladora del robot (*ROBOT -> PINZA*).

En la esquina inferior izquierda, podemos apreciar los zócalos para colocar un display de 7 segmentos, junto con su resistencia limitadora de intensidad. Y en la esquina inferior derecha va alojado un potenciómetro, un led RGB y un zumbador.

Por último, la placa de *Arduino Mega* ha sido insertada por debajo de la PCB, de modo que para proceder a su conexión, solo hay que clavar directamente la PCB sobre el *Arduino*.

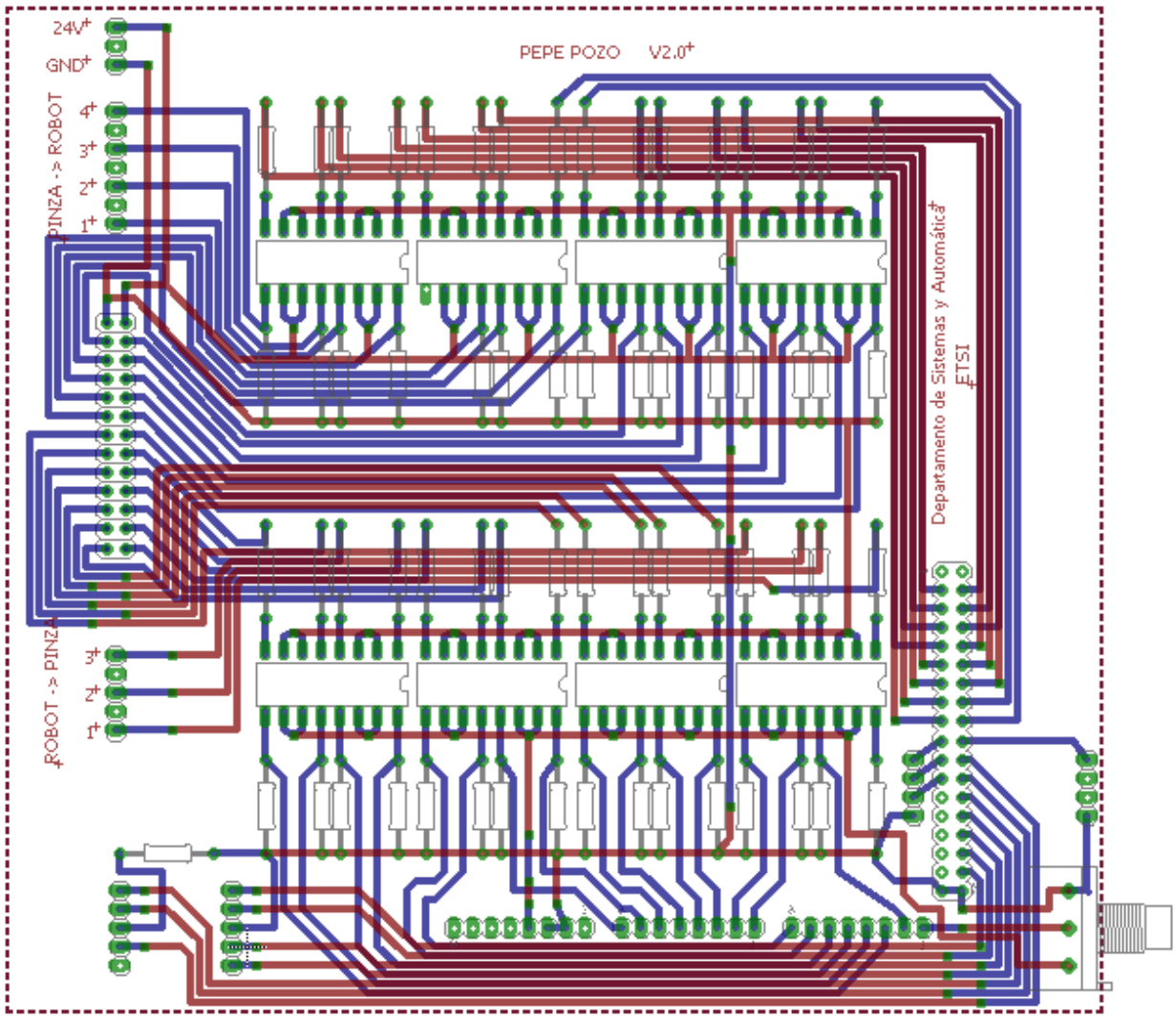


Figura 9: TOP y BOTTOM de la PCB

3.4 Fabricación de la PCB

3.4.1 Fotolito

El fotolito es la transparencia con el esquema eléctrico impreso que nos permite imprimir las zonas que necesitemos y dejar el resto en blanco para que así la luz UV pueda traspasarlo. Debe ser impreso en papel tranparente mediante impresora laser para asegurar la calidad de la insolación.

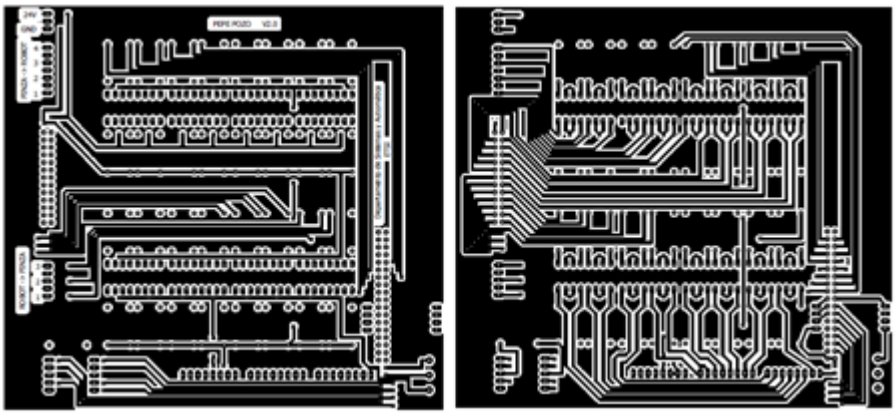


Figura 10: Fotolito (Top & Bottom)

3.4.2 Preparación de la placa

Lo primero que debemos hacer es recortar la placa de fibra de vidrio o baquelita que necesitemos, para ello tomamos medidas y cortamos haciendo uso de una gillotina. Una vez cortada la placa deberemos de limar los laterales con una lija de grano fino para eliminar las imperfecciones.

3.4.3 Insolación

Con la placa y el fotolitos preparados ya podremos pasar al insolado, para ello emparejamos la placa y el fotolito en la posición correcta, los ponemos sobre el cristal y cerramos la tapa para que comience el insolado. El proceso dura unos 3 minutos.

Figura 11: Fotolito y placa en la insoladora

3.4.4 Revelado

El revelado se compone de 1 litro de agua templada y unos 12 gramos de sosa caustica. Sumergimos la placa con cuidado de no rayarla. El tiempo de revelado se comprende entre 30 y 60 segundos, veremos que el barniz expuesto a la luz UV comienza a oscurecerse y poco después a desprenderse. Tenemos que tener cuidado de no exponer la placa en exceso a esta solución o el resto de barniz podría comenzar a desprenderse, si esto ocurriese nos quedaría una placa inservible. Al terminar el revelado lavaremos la placa con abundante agua y ya la tendremos lista para el atacado.

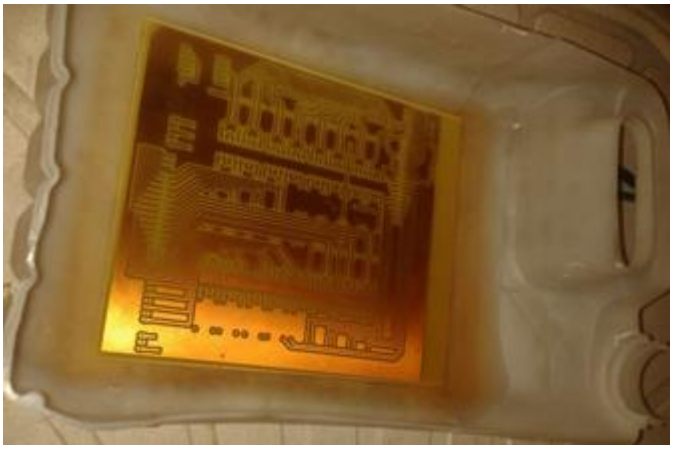


Figura 12: Inmersión en líquido revelador

3.4.5 Atacado

El atacado se encarga de corroer el cobre que está desprotegido. Se sumerge la placa en una mezcla de agua, ácido clorhídrico y dióxido de hidrógeno (2:1:1). Una vez se desprenda todo el cobre y queden las pistas bien definidas podremos sacar la placa para enjuagarla con abundante agua.

3.4.6 Taladrado

Mediante un taladro vertical, perforamos los orificios para los zocalos, vías, resistencias, y demás componentes.

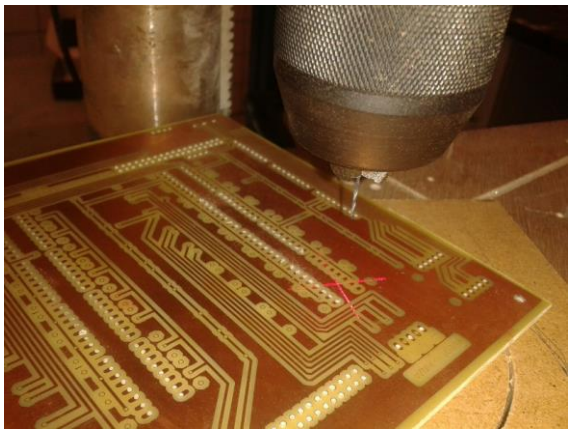


Figura 13: Taladrado

3.4.7 Soldadura de vías, resistencias y zócalos

Se sueldan las vías haciendo pasar un cable por los orificios y añadiendo estaño en ambas caras. Se insertan las resistencias y los zócalos para los componentes.

3.4.8 Inserción de componentes

Se insertan los optoacopladores en sus bases, así como los leds, el indicador 7 segmentos, el buzzer y el *Arduino Mega*, quedando las conexiones como se muestra en la siguiente imagen:

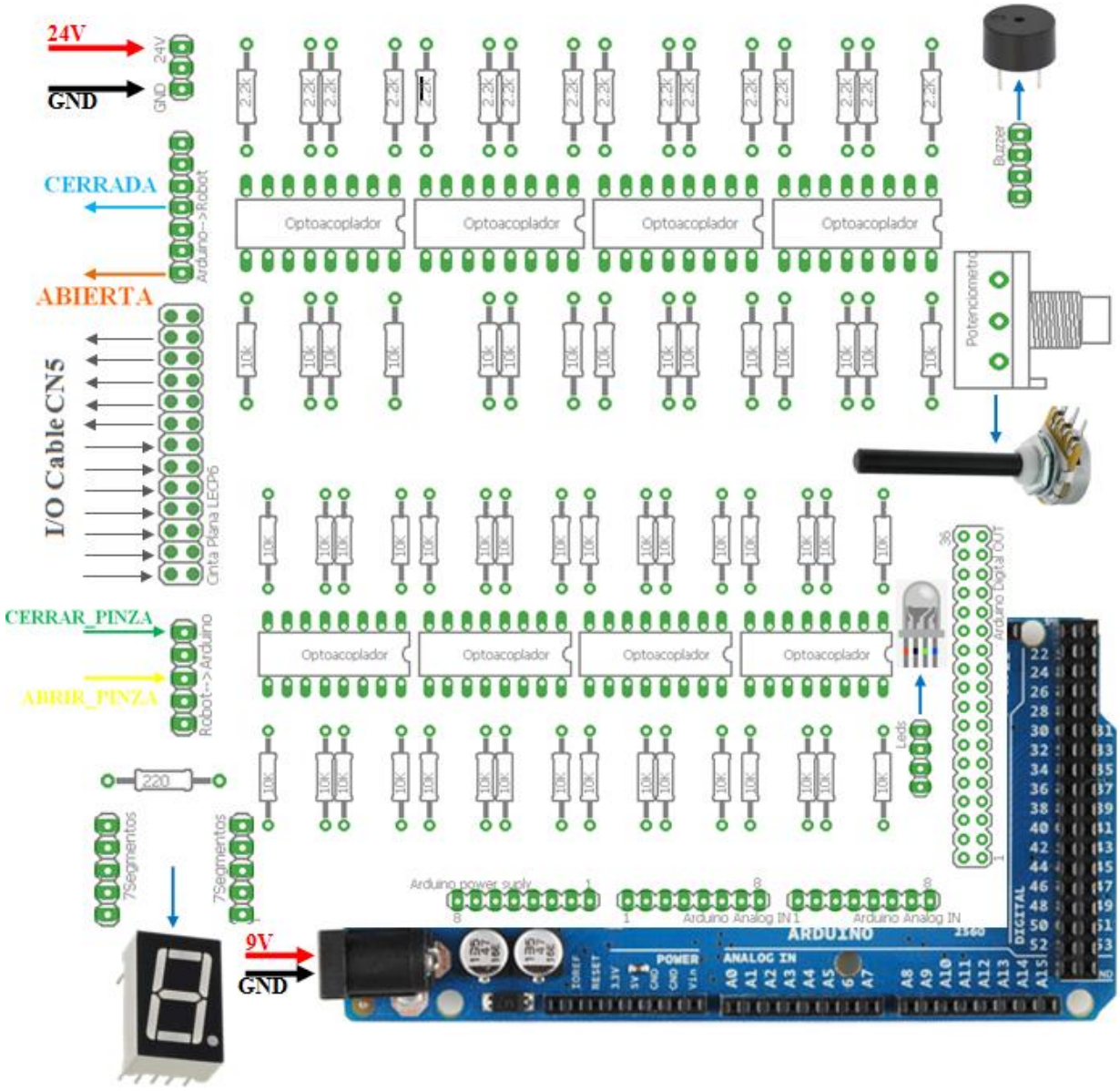
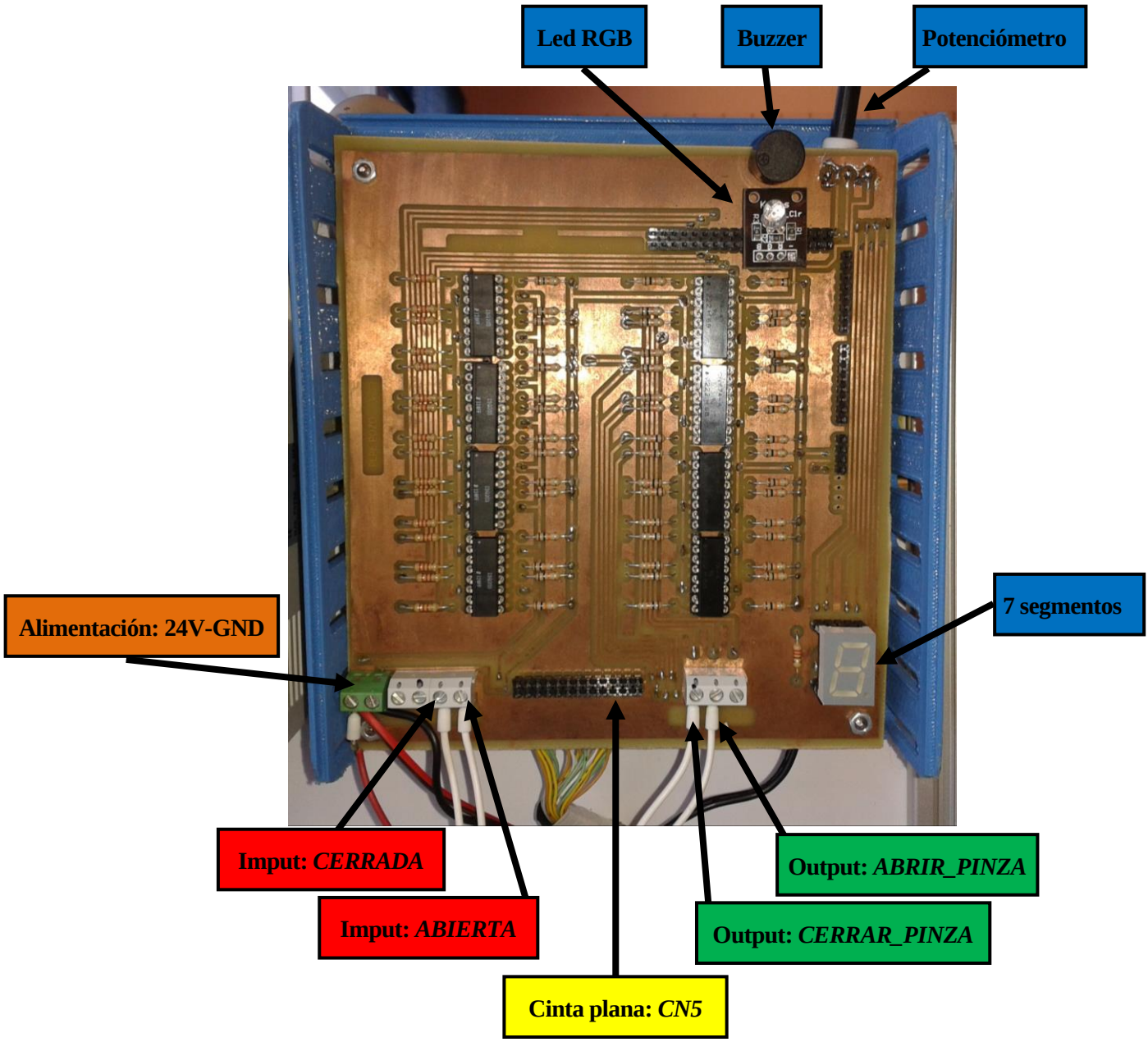


Figura 14: Cableado e inserción de componentes

En la parte posterior de la placa clavamos en *Arduino Mega*.

Figura 15: Parte trasera de la placa



3.5 Montaje de PCB, controlador LEC-P6 y fuentes de alimentación.

El último paso consiste en colocar los equipos sobre un panel anclado a un carril DIN que soporte los sistemas. El interruptor bipolar cortará la alimentación de las fuentes de 24V y de 9V como indica el esquema.

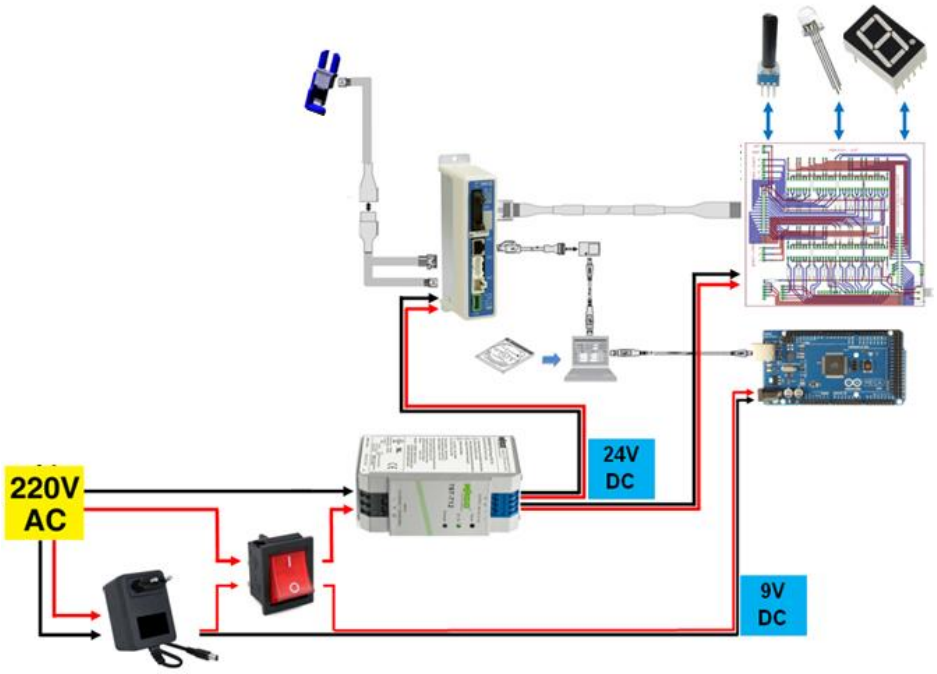


Figura 16: Esquema de conexiones del panel

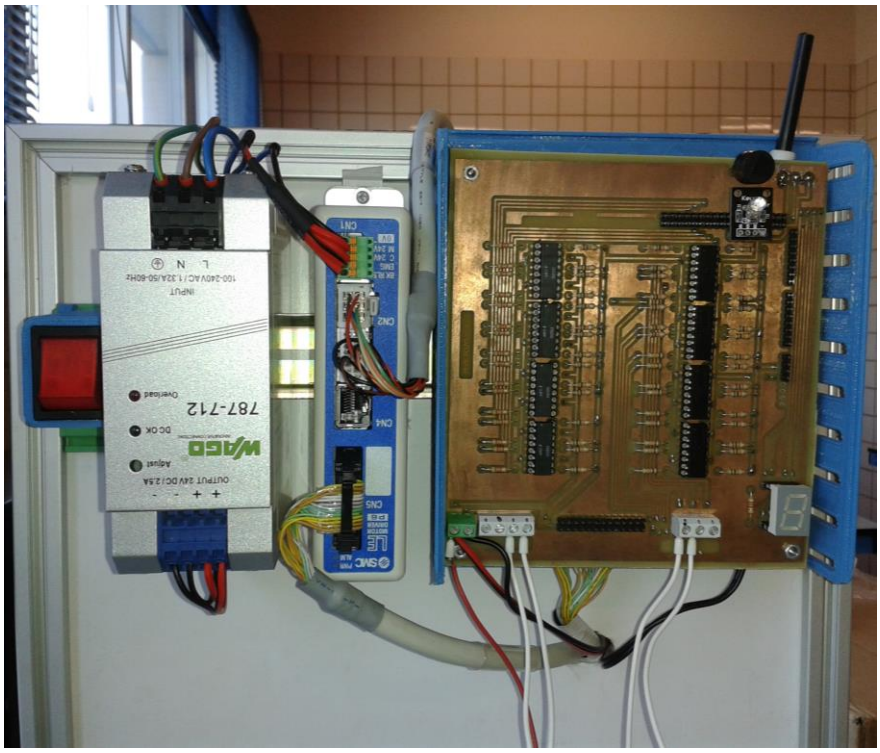
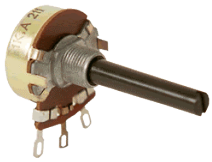
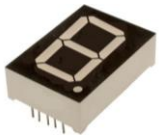


Figura 17: Panel de componentes

3.6 Lista de componentes

A continuación se detallan el número mínimo de componentes necesarios para llevar a cabo la fabricación de la PCB.

Componente	Cantidad	Imagen
Placa para circuito impreso	1	
Optoacopladores *CNY74-4H* ***Ver Nota***	8	
Resistencias 10KΩ	64	
Resistencia 220Ω	1	
Potenciómetro	1	
Buzzer	1	
Led RGB	1	
7 segmentos	1	

PCB bloques terminales 2 pin	3	
PCB bloques terminales 3 pin	1	
Tira de pines macho 2.54mm 18mm de largo	86 pines	
Tira de pines hembra 2.54mm	146 pines	

Tabla 10: Lista de componentes para la PCB

NOTA:

Para la realización de este trabajo, se han empleado dos tipos de optoacopladores, en concreto los modelos *ISQ-201* y *CNY74-4H.* Actualmente, el modelo *ISQ-201* se encuentra descatalogado. Si se quisiera volver a fabricar esta placa, habría que usar 8 integrados modelo *CNY74-4H,* los cuales se encuentran aún en el mercado. Es por ello por lo que, en la lista de componentes anterior, solo se encuentra el modelo *CNY74-4H,* así como las resistencias necesarias para estos 8 dispositivos. Si se quisiera fabricar la placa de nuevo, totas las resistencias de dicha placa serían de 10K, tal y como se encuentra la línea de 4 optoacopladores modelo *CNY74-4H.*

La siguiente imagen ayuda a aclarar las operaciones de inserción y soldadura. Las vías no aparecen representadas. Por definición, su soldadura es trasera y delantera.

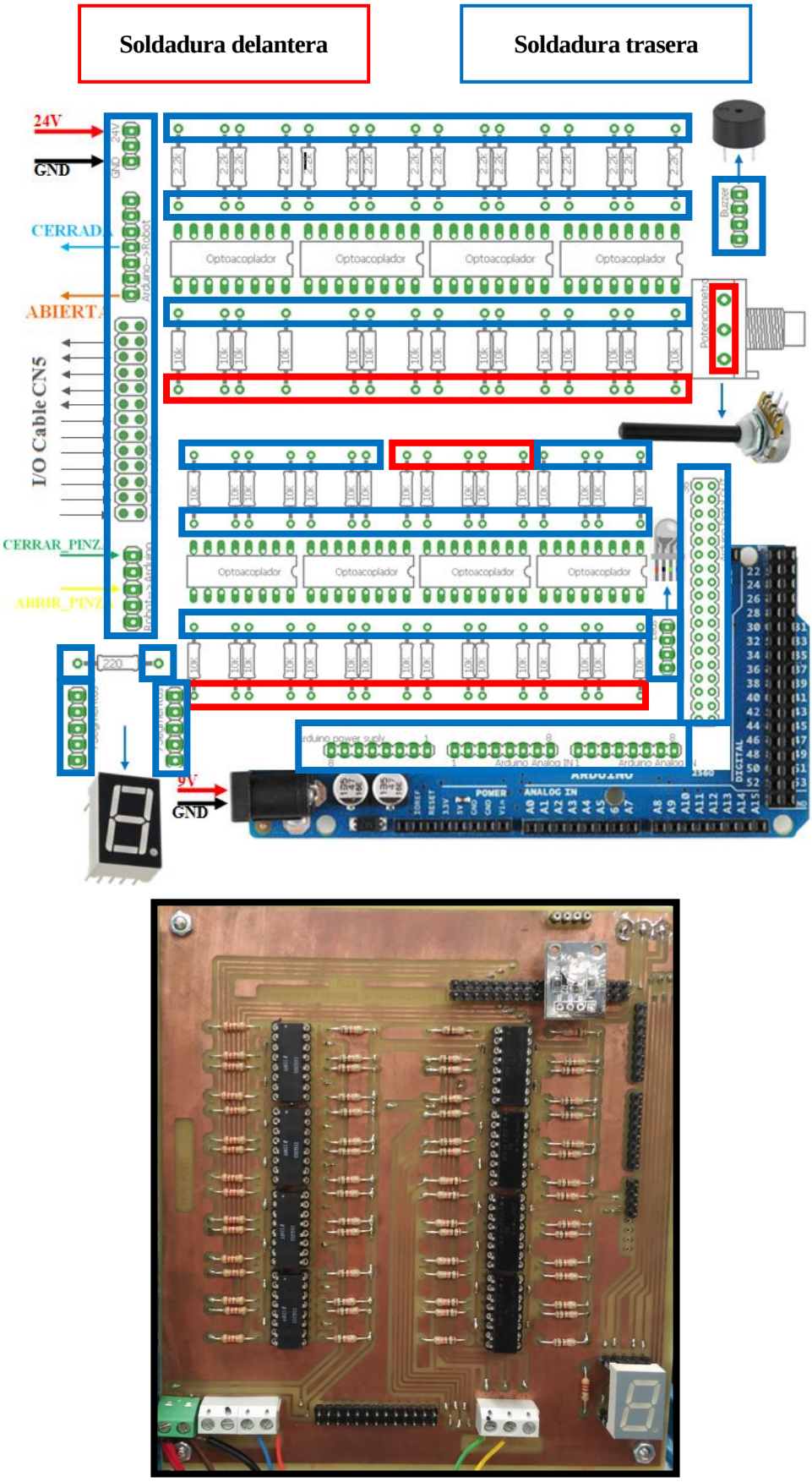


Figura 18: Soldaduras delanteras y traseras de la PCB

4 PROGRAMACIÓN DE ARDUINO Y LEC-P6

4.1 Introducción

Para programar el funcionamiento de Arduino vamos a utilizar la herramienta *StateFlow* de *Matlab-Simulink*.desde **Matlab 2013b.** Dicha librería nos permite crear una maquina de estado que controle la comunicación entra la controladora del robot y la pinza.

Es importante recalcar que **Arduino por sí solo no se encarga del control en bajo nivel** de la pinza. Él solo ejecuta órdenes de alto nivel que previamente deben haber sido programadas en el driver LEC-P6. Primero abordaremos las órdenes de alto nivel suministradas por Arduino para, a continuación, detallar como se programan las posiciones en el driver LEC-P6 de la pinza.

4.2 Arduino en MatLab-Simulink

En el siguiente enlace, se muestran los pasos necesarios para instalar los paquetes de soporte y se describen los modos de ejecución posible. Desde *Matlab*, es posible controlar *Arduino* y utilizarla como una placa de E/S, pudiendo obtener información de los sensores conectados a la placa y actuando sobre diferentes actuadores. Toda la ejecución del algoritmo ocurre en MATLAB. Desde Simulink, podemos desarrollar un modelo, utilizando la biblioteca de bloques del paquete de soporte de Simulink para Arduino, que se ejecute de modo embebido en el procesador de la placa Arduino.

Instalación y programacion de Arduino con Simulink

4.3 Stateflow

Stateflow es una librera (toolbox) de *Matlab* que permite modelar sistemas de eventos discretos dentro de *Simulink*, utilizando máquinas de estado.

4.3.1 Estados

Representan estados del sistema de eventos discretos. Junto a la esquina superior izquierda, cada rectángulo lleva un texto con un nombre que identifica al estado. La sintaxis de Stateflow permite especificar el instante en que se iniciará la acción y la duración de esta:

- ***entry*:** la acción se inicia al entrar en este estado.
- ***exit*:** la acción se inicia al salir de este estado.

4.3.2 Transiciones

Las transiciones tienen forma de flecha y representan saltos entre estados, asociados a eventos, que se producen en el sistema. Si el sistema está en el estado 1 y se produce el evento e, entonces el sistema pasa al estado 2. Una transición especial es la llamada transición por defecto (*Default-transition*), que sirve para señalar el estado inicial del sistema. Se reconoce por su forma ya que en el extremo opuesto a la derecha lleva un pequeño círculo Negro. Cada transición puede tener un texto escrito junto a ella que indica el evento que ha de producirse para que se dispare la transición.

- **[C]** (en donde c es una condición): la transición se dispara sí la condición c (expresión booleana) es verdadera.

4.4 Programa de prueba para verificar señales

El primer programa que se ha diseñado no cuenta con ninguna máquina de estados que gestione entradas o salidas. En lugar de ello, consiste en un bucle abierto donde las señales de entrada son visualizadas, y las señales de salida pueden ser activadas manualmente.

La finalidad de este simple programa es la de verificar que la comunicación entre sistemas se está efectuando correctamente, que no hay fallos de conexionado en el hardware utilizado, y que los niveles necesarios de tensión a la entrada y salida de los dispositivos son los adecuados.

Como se puede observar en la siguiente imagen, el programa cuenta con dos susbsistemas, uno para leer las variables de entrada, y otro para activar las variables de salida.

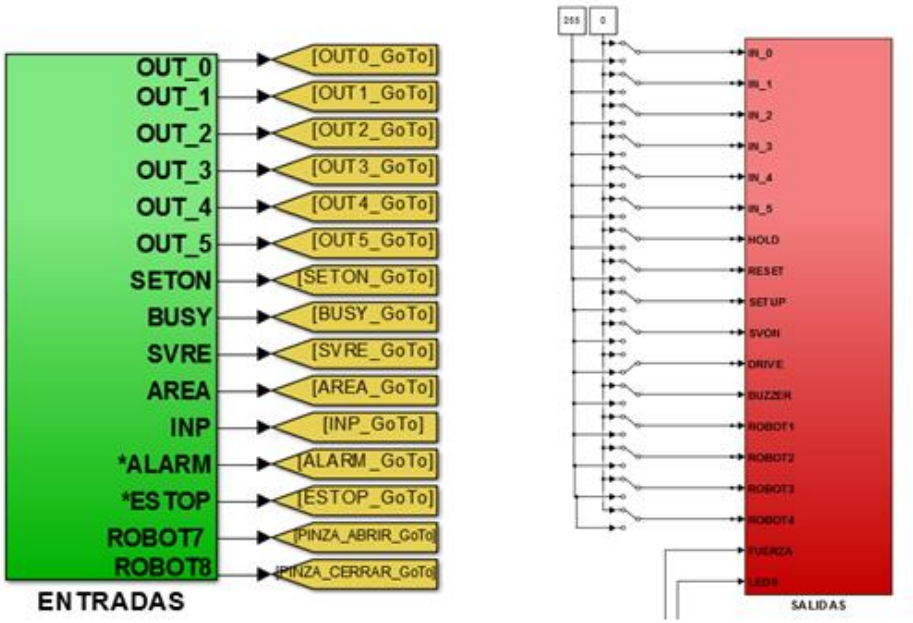
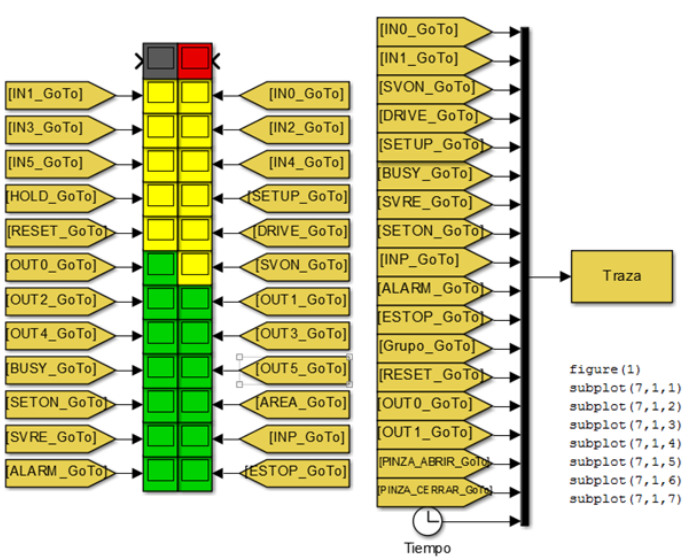


Figura 19: Programa de pruebas

La visualización de las señales puede efectuarse de dos formas. La primera, mediante el uso de *displays* para poder controlar el estado de la señal en tiempo real. Y la segunda, consiste en grabar todas las señales en el *Workspace* de Matlab para luego representarlas gráficamente mediante el script llamado ***Traza***.




```
% --  Programa para representar el Log obtenido
% --    de la ejecución del programa Pinza
close all

IN0=Traza(:,1);
IN1=Traza(:,2);
SVON=Traza(:,3);
DRIVE=Traza(:,4);
SETUP=Traza(:,5);
BUSY=Traza(:,6);
SVRE=Traza(:,7);
SETON=Traza(:,8);
INP=Traza(:,9);
```

```
figure(1)
subplot(7,1,1),  plot(Tiempo,IN0,'r','LineWidth',4);  ylabel('IN0');
subplot(7,1,2),  plot(Tiempo,IN1,'r','LineWidth',4);  ylabel('IN1');
subplot(7,1,3),  plot(Tiempo,OUT0,'b','LineWidth',4);  ylabel('OUT0');
subplot(7,1,4),  plot(Tiempo,OUT1,'b','LineWidth',4);  ylabel('OUT1');
subplot(7,1,5),  plot(Tiempo,ALARM,'b','LineWidth',4); ylabel('ALARM');
subplot(7,1,6),  plot(Tiempo,PINZA_CERRAR,'b','LineWidth',4);  ylabel('P_CERRAR');
subplot(7,1,7),  plot(Tiempo,PINZA_ABRIR,'b','LineWidth',4);  ylabel('P_ABRIR');
```

Figura 20: Log de datos y representaciones gráficas

4.5 Programa PINZA

El programa Pinza se compone de los subsistemas de lectura de entradas, la máquina de estados, y la escritura de las salidas.

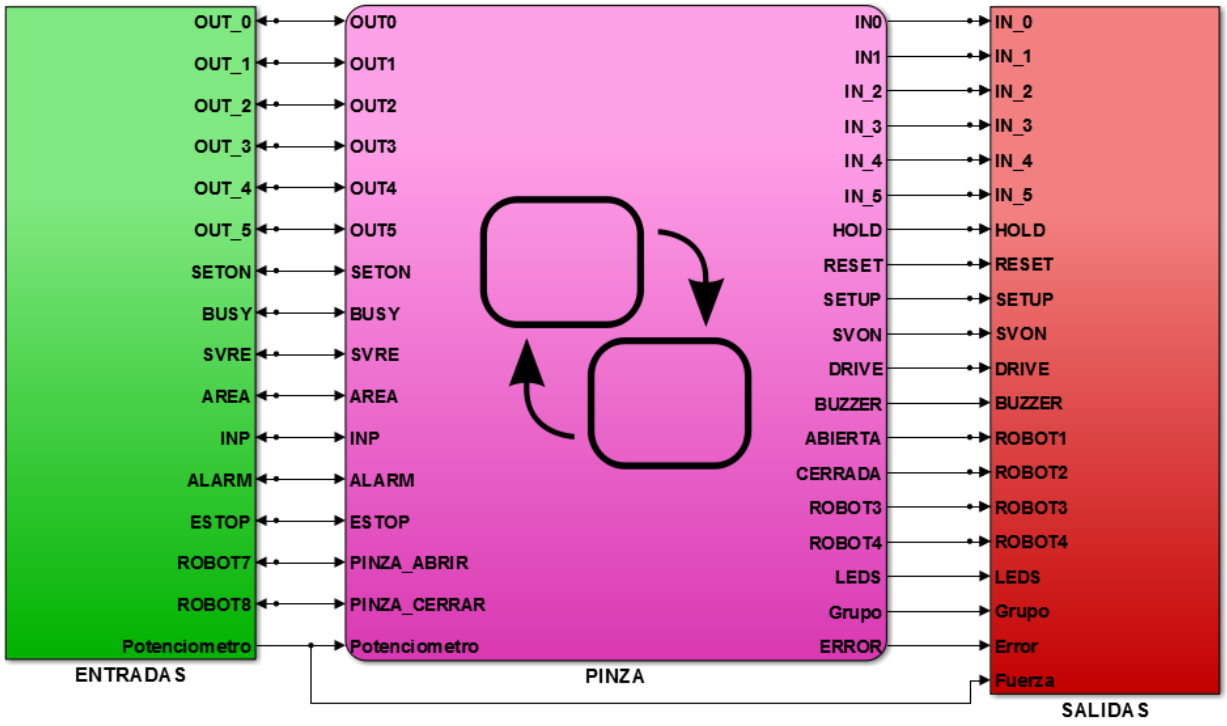


Figura 21: Lectura, escritura de señales, y máquina de estados

4.5.1 Entradas

El subsistema que agrupa las entradas está compuesto por bloques de *Arduino* de entradas analógicas. A cada una de ellas se le asigna su nombre correspondiente. El valor de dichas entradas varía de 0 a 1024. Mediante los bloques de condición, saturamos dicho a valor a los niveles lógicos 0 o 1.

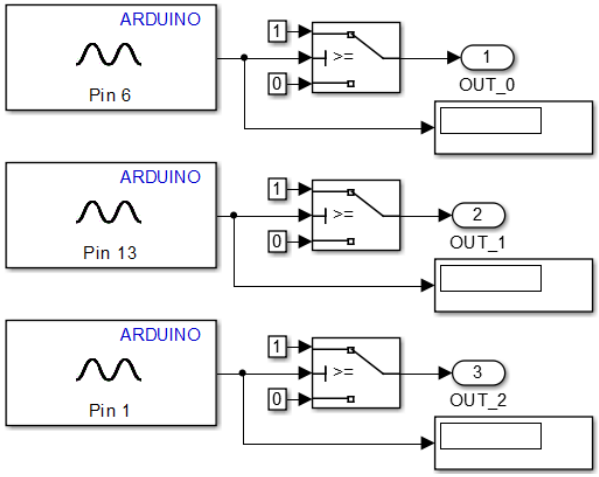


Figura 22: Lectura de señales analógicas

Una de esas señales analógicas corresponde al valor de medida del potenciómetro instalado en la PCB. Dicho valor analógico se hace pasar por una *Matlab function* que traducirá el rango de 0-1024 a cinco niveles enteros distintos (0-5). Más adelante, se explicará que estos niveles corresponden a la fuerza de operación seleccionada para la pinza.

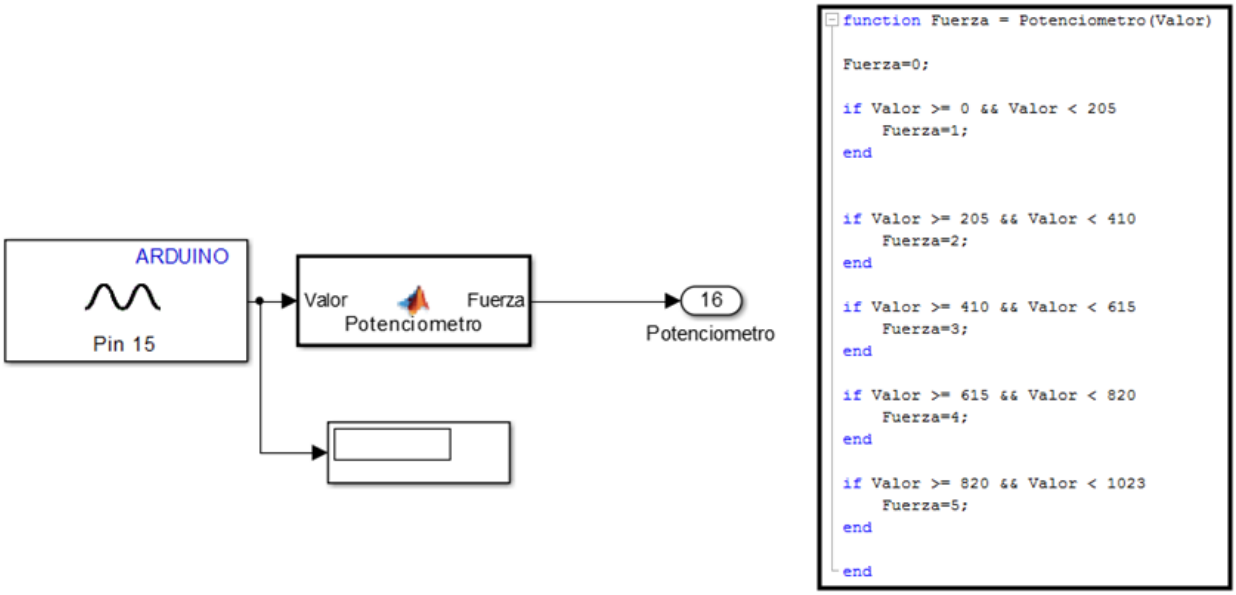


Figura 23: Potenciómetro

4.5.2 Salidas

El subsistema salidas, agrupa la escritura de señales de salida digitales mediante bloque de *Arduino*.

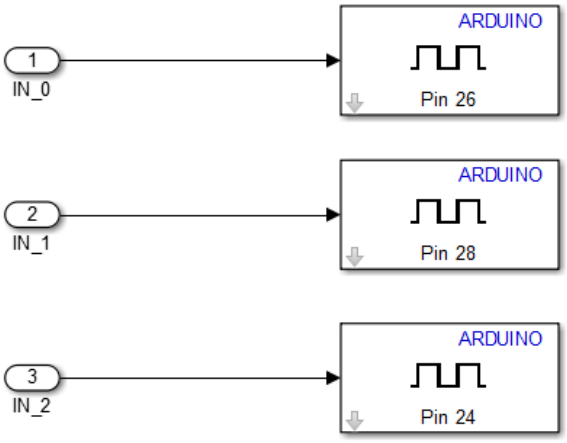


Figura 24: Escritura de señales digitales

Las señales de salida se componen de las ordenes de entrada al controlador LEC-P6 de la pinza, pero además, también se gestiona un *buzzer*, un led RGB y un display de 7 segmentos. La señal de control del *buzzer* consiste simplemente en una señal digital de salida (0-1). El led RGB se compone de tres diodos luminosos que se activan mediante señales digitales. Sin embargo, el estado de estas salidas dependerá de si nos encontramos en *Modo Error* o *Modo Normal*.

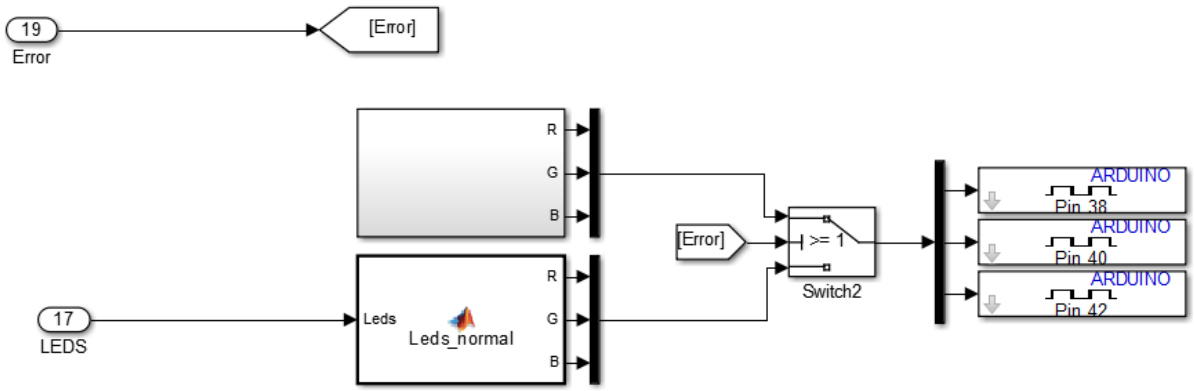


Figura 25: Control del led RGB

Si el sistema se encuentra en *Modo Normal*, el estado de los pines 38,40 y 42 (R,G,B) será el que dictamine la *Matlab function*: **Leds_normal.** Por el contrario, si nos encontramos en *Modo Error*, el control conmutará hacia el subsistema superior, con lo que simplemente se lleva a cabo el parpadeo del led rojo.

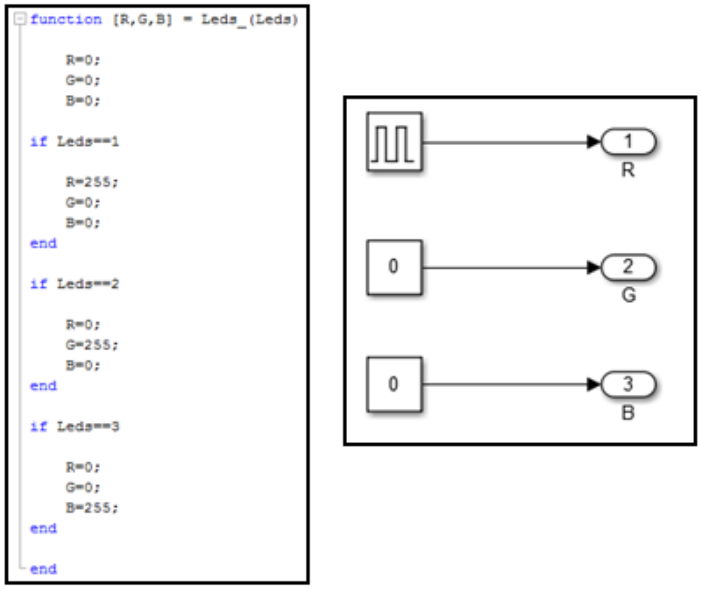


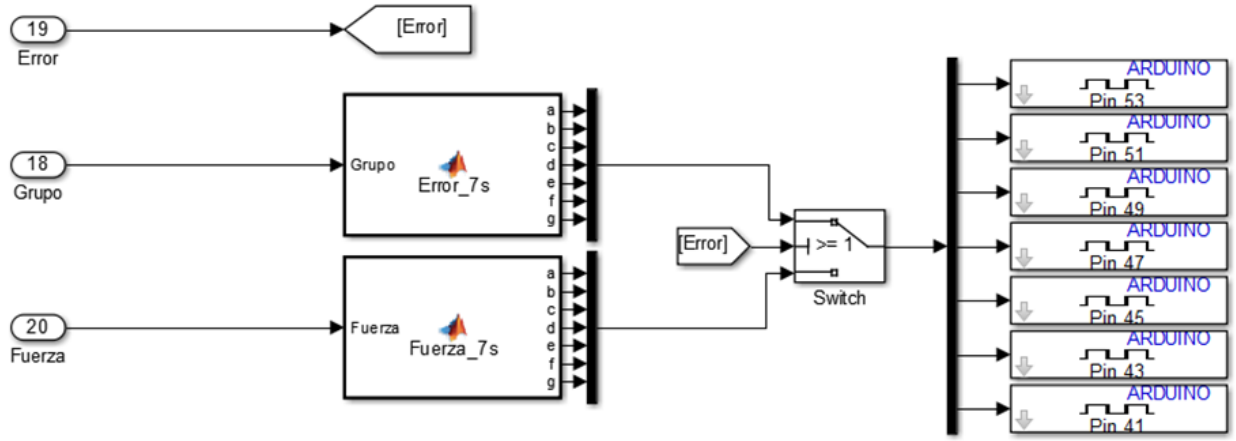
Figura 26: Modo Normal – Modo Error

Significado del indicador RGB		
ROJO	**Al inicio**	No se deben enviar órdenes. El sistema está iniciándose.
	Durante el funcionamiento normal	Error. El sistema se reiniciará y el display 7 segmentos indica el grupo (tipo) de error.
AZUL	**Durante el funcionamiento normal**	No se deben enviar órdenes. La pinza se está moviendo a la posición indicada previamente.
VERDE	**Durante el funcionamiento normal**	El sistema está a la espera de recibir órdenes. La pinza está en reposo (abierta) o está a la espera haciendo presión contra un objeto.

Tabla 11: Estados del led RGB

Por último, solo queda el control del display 7 segmentos. Este también dependerá de si el sistema se encuentra en *Modo Normal / Modo Error*. En el caso del *Modo Normal*, el display indica la fuerza que se ha ajustado en el potenciómetro. Por el contrario, en el *Modo Error*, el display visualiza al operado el "*Grupo* (tipo) *de Error*" que el controlador está indicando.

Para obtener más información sobre los "Grupos de Error" consultar la página 51, **Alarm Detection,** del Manual de Operación LEC-P6.




```
function [a,b,c,d,e,f,g] = Fuerza_7s(Fuerza)

    a=0;
    b=255;
    c=255;
    d=255;
    e=255;
    f=255;
    g=255;

if Fuerza==1

    a=0;
    b=0;
    c=0;
    d=255;
    e=0;
    f=0;
    g=255;
end
```

```
function [a,b,c,d,e,f,g] = Error_7s(Grupo)

    a=0;
    b=255;
    c=255;
    d=255;
    e=255;
    f=255;
    g=255;

if Grupo==1

    a=0;
    b=0;
    c=0;
    d=255;
    e=0;
    f=0;
    g=255;
end
```

Figura 27: Control del display 7 segmentos

Significado del display 7 segmentos		
FUNCIONAMIENTO NORMAL	**1-5**	Fuerza de empuje de la pinza, ajustada por el potenciómetro.
MODO ERROR	**1-4**	Grupo (tipo) de error.

Tabla 12: Estados del display 7 segmentos

4.5.3 Pinza

La máquina de estados *Pinza* sigue las pautas de la guía GEMMA (*Guide d'Etude des Modes de Marches et d'Arrêts*, Guía de estudio de los modos de marchas y paradas).

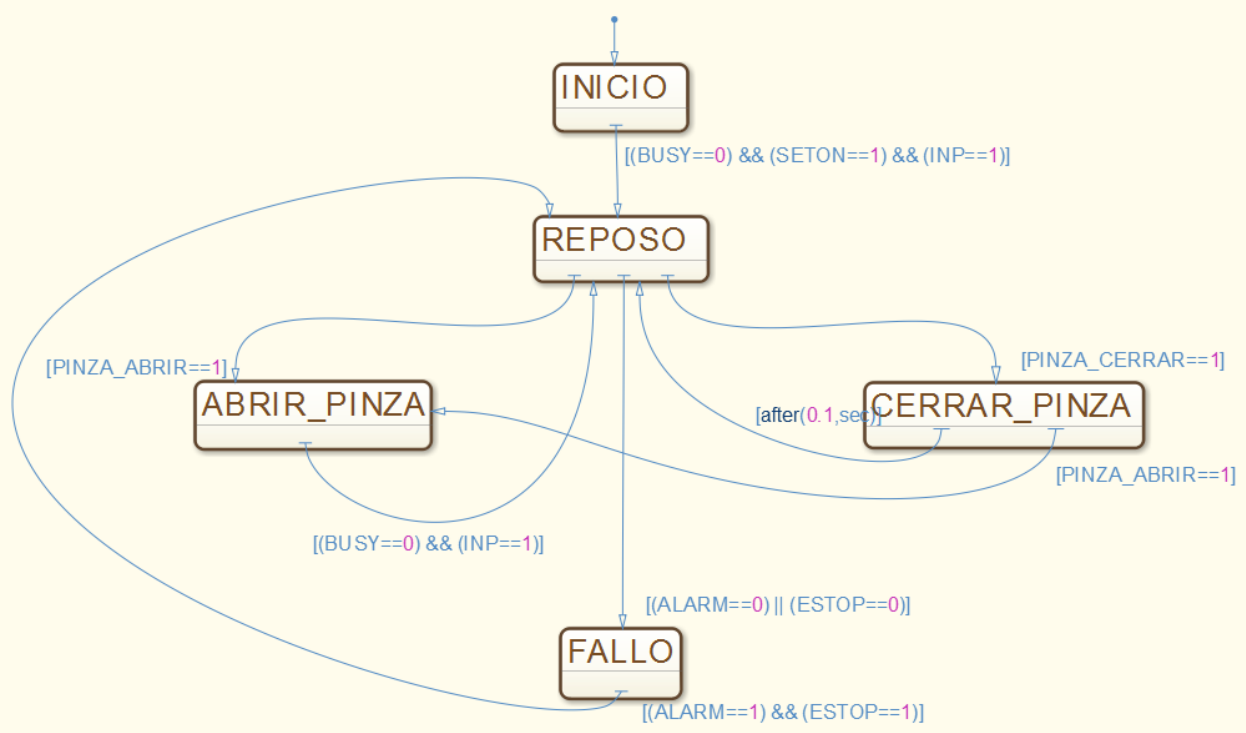


Figura 28: Máquina de estados

El GRAFCET se compone de los estados que pueden visualizarse en la imagen anterior. A continuación se irán detallando uno a uno. Para más información sobre las variables del controlador LEC-P6 consulte:

Anexo E: Variables – Controlador LEC-P6

Manual de Operación LEC-P6

4.5.3.1 Inicio

El estado *Inicio* es siempre el primero en ejecutarse cuando el sistema se pone en funcionamiento, dado que la transición por defecto (*Default-transition)* descansa sobre él. Cuando se activa, la variable *LEDS* se inicializa al valor 1. Ello implica que a la función *Leds_normal* le está llegando un 1, por lo que el led RGB permanecerá rojo hasta que el sistema se inicialice. Mediante SVON, damos alimentación a los servos de la pinza. Mientras tanto, el *buzzer* indica sonoramente que la pinza de está inicializando. A continuación, activamos la señal *SETUP* para enviar la pinza a su origen. El led RGB se torna azul indicando que el sistema está ocupado ejecutando órdenes. Cuando la pinza ha retornado al origen, se activan las señales de salida *CERRADA*, y se desactiva la señal *ABIERTA* que le indicaran a la controladora del robor *IRC-5* que la pinza está cerrada (porque esa es su posición de origen). El led RGB se vuelve verde (esperando nuevas ordenes) y el buzzer se apaga. Para salir del estado de Inicio, el controlador debe enviarnos las señales *BUSY=0* (significa que la pinza ya no esta ocupada moviendose), *SETON=1* (indica que el retorno al origen solicitado mediante *SETUP* se ha completado) e *INP=1* (que la pinza se halla en la posición deseada).

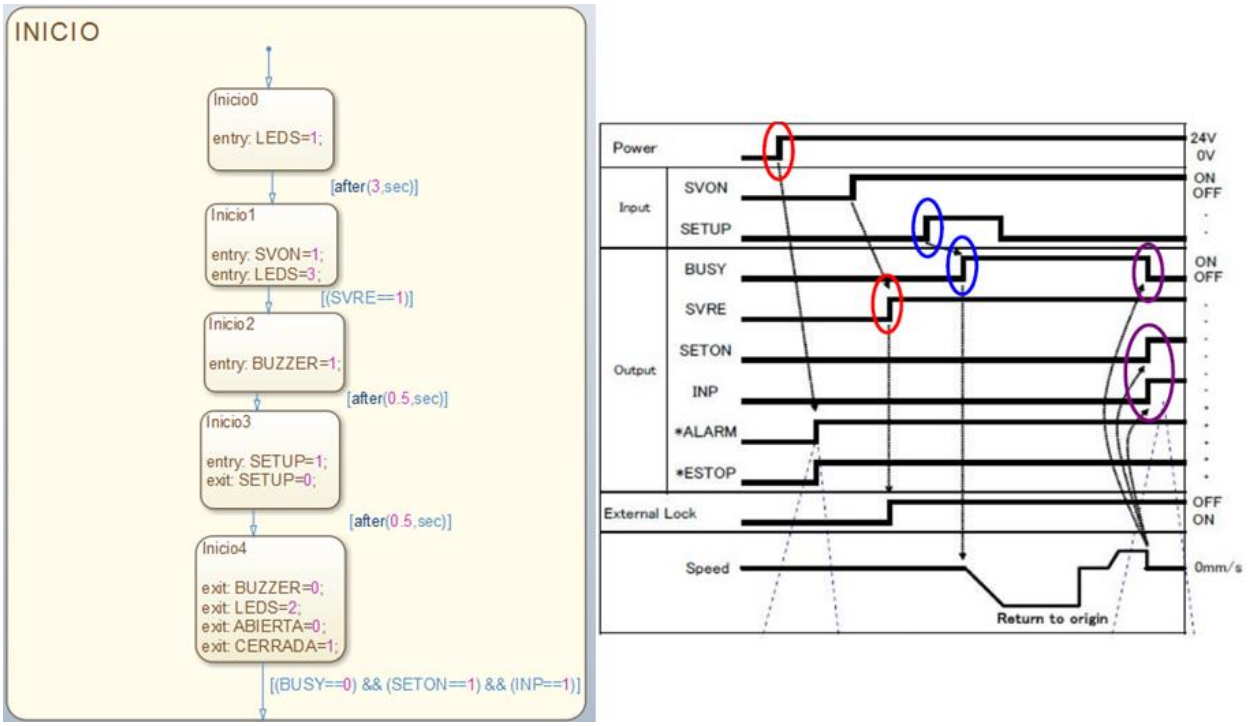


Figura 29: Estado inicio

4.5.3.2 Reposo

Cuando el estado *Inicio* finaliza, se pasa automáticamente al estado *Reposo*. En dicho estado, se corta la alimentación a los servo motores (*SVON=0*), se apaga el *buzzer*, y el led marca el color verde (a la espera). Además, las variables *IN*, usadas para definir las ordenes de movimiento deseasas, de resetean a 0.

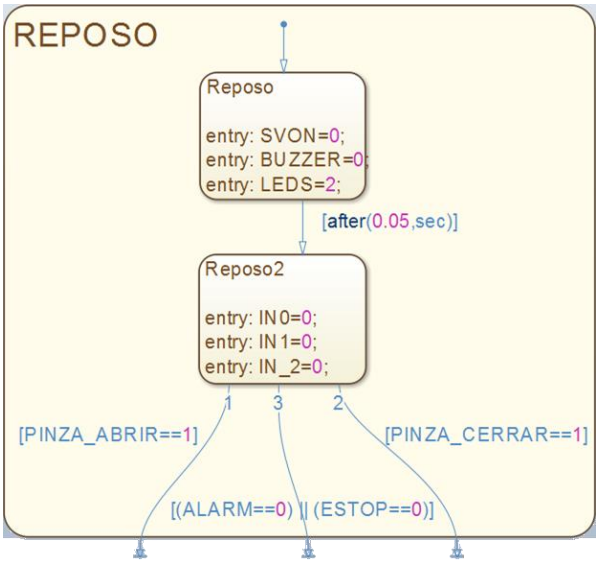


Figura 30: Estado reposo

4.5.3.3 Abrir Pinza

Desde el controlador del robot IRC-5 se activan las señales *PINZA_ABRIR* y *PINZA_CERRAR*. Cuando la máquina de estados recibe la señal PINZA_ABRIR, se pasa del estado *Reposo* al estado *Abrir Pinza*. Lo primero que se hace es reactivar los servos y señalizar mediante led azul que el sistema está ocupado. Se activa la señal IN0 y se mantiene a 0 la señal IN1. Esta codificación señala que se quiere ejecutar la orden 1.

0:OFF 1:ON

Step No	IN5	IN4	IN3	IN2	IN1	IN0
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	0	0	1	0	0
5	0	0	0	1	0	1

Tabla 13: Ordenes (Step) que pueden ser demandadas

Una vez que el driver ha leído la orden que le hemos enviado (Orden 1), la cual corresponde a una operación de posicionado que deja pinza abierta, activamos la señal *DRIVE* que hace que la pinza ejecute dicha orden. Las condiciones de finalización del estado exigen que la pinza ya no esté moviéndose (*BUSY=0*) y que haya alcanzado la posición deseada (*INP=1*).

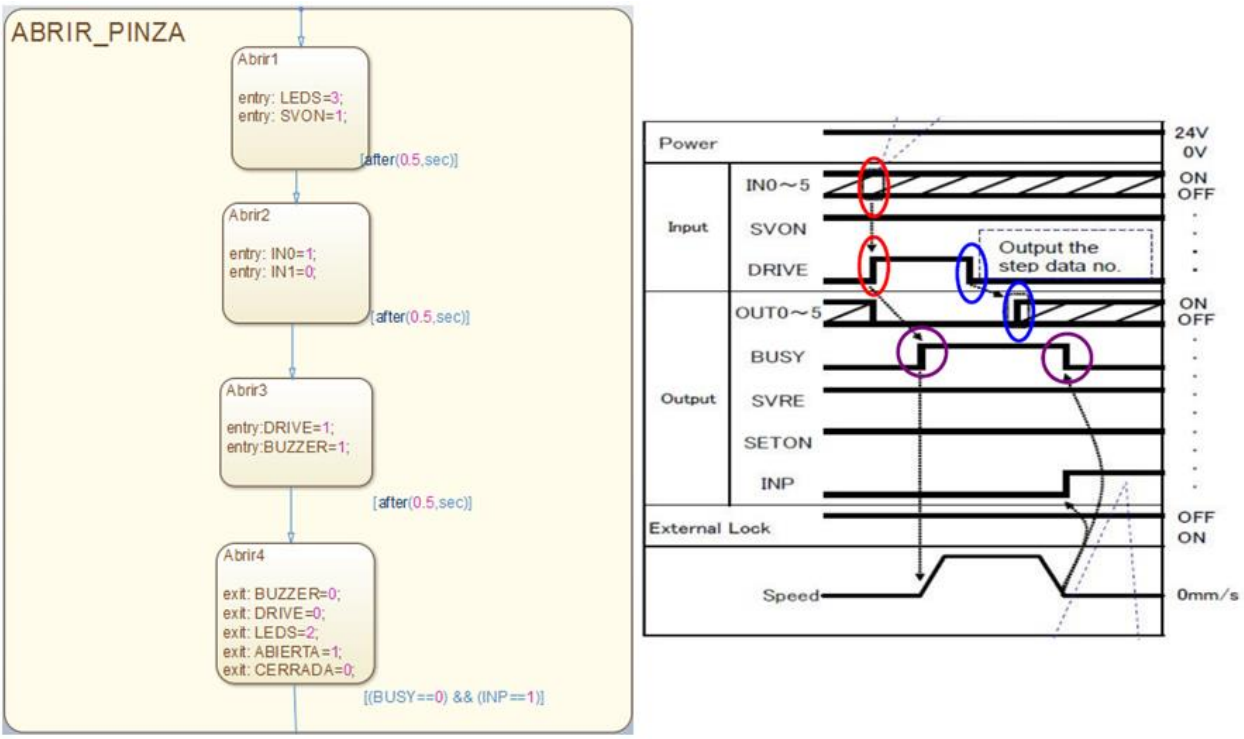


Figura 31: Estado Abrir Pinza

4.5.3.4 Cerrar Pinza

Para cerrar la pinza, el controlador del robot debe ejecutar la orden *PINZA_CERRAR*. En ese momento, se activará el estado *Cerrar Pinza*, el cual difiere del anterior en un par de detalles. En primer lugar, la selección de la orden (Step) vendrá condicionada por el valor entero (1-5) que marque en ese momento el potenciómetro. Por ello, dependiendo de la posición del potenciómetro, se enviará una orden u otra. Las órdenes enviadas ya no desencadenan una maniobra de posicionamiento absoluto. En lugar de ello, la pinza efectuará un empuje con una fuerza proporcional al valor del potenciómetro.

La operación de control de fuerza puede finalizar de dos maneras. En la primera de ellas, no hay objeto alguno entre los dedos de la pinza. Se ha estado empujando contra el aire sin resistencia alguna. Cuando se alcanza el final de la carrera de la pinza, se pasa al subestado *AIRE*, en el cual simplemente se activa la señal CERRADA, comunicando al controlador del robot que la pinza está cerrada, y se pasa al estado de reposo, en el cual se apaga el servo.

Por otro lado, si se empuja contra un objeto, dicho objeto será comprimido con la fuerza deseada. Cuando esa fuerza se alcance, el sistema pasará al subestado *OBJETO*, en el cual se debe permanecer sin apagar el driver, pues la pinza debe seguir empujando con la misma fuerza para evitar que el objeto agarrado caiga. Solo la orden PINZA_ABRIR puede sacar al sistema de este estado, por lo que solo cuando la controladora del robot lo demande, la pinza se relajará volviendo al estado Abrir Pinza.

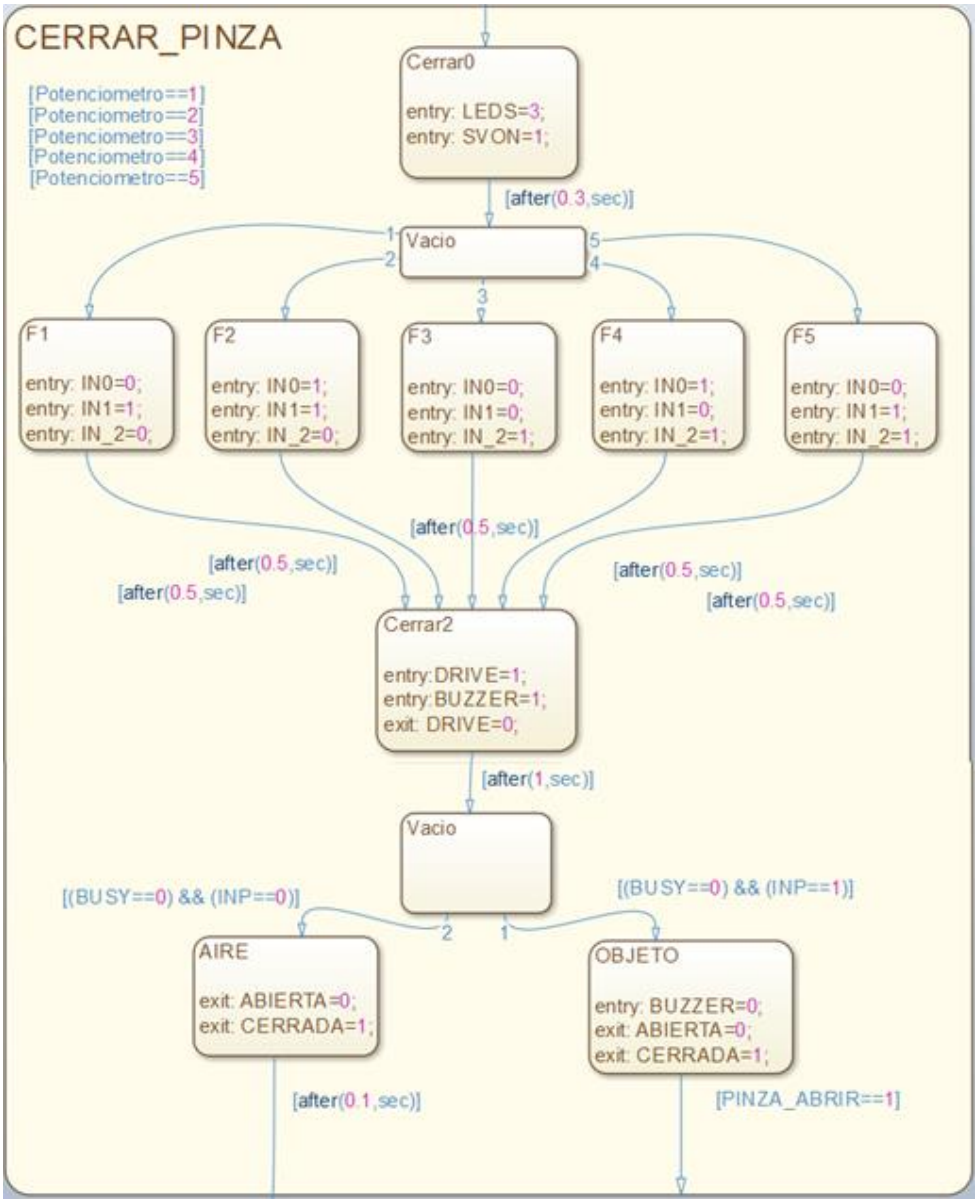
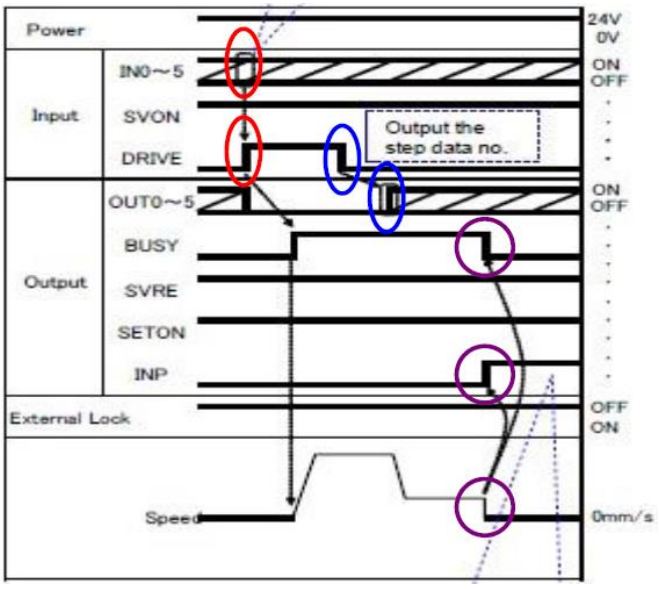


Figura 32: Estado Cerrar Pinza



4.5.3.5 Fallo

Durante el funcionamiento normal, las señales provenientes del driver LEC-P6 denominadas *ALARM* y *ESTOP*, permanecen con un valor lógico alto, es decir, a 1. Estas dos señales trabajan a modo de cero vivo, es decir, si ocurriera algún error o se produjera una desconexión entre el driver y *Arduino*, estas señales caerían a 0, y se detectaría que algún fallo está sucediendo. Si eso ocurriera, automáticamente se activaría el estado *Fallo*. En dicha situación, lo primero que se hace es apagar el servo. Acto seguido, se activa la variable interna *Error*, la cual provoca que el led rgb se torne a rojo, y el display 7 segmentos marque el error que indica la variable *Grupo*. La decisión de qué grupo activar, la determinan el conjunto de señales *OUT* las cuales cambiaran dependiendo del tipo de fallo producido.

El último paso es resetear el driver activando la señal *RESET*.

0:OFF 1:ON

Step No	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	0	0	1	0	0

Tabla 14: Señales OUT que marcan el tipo de fallo

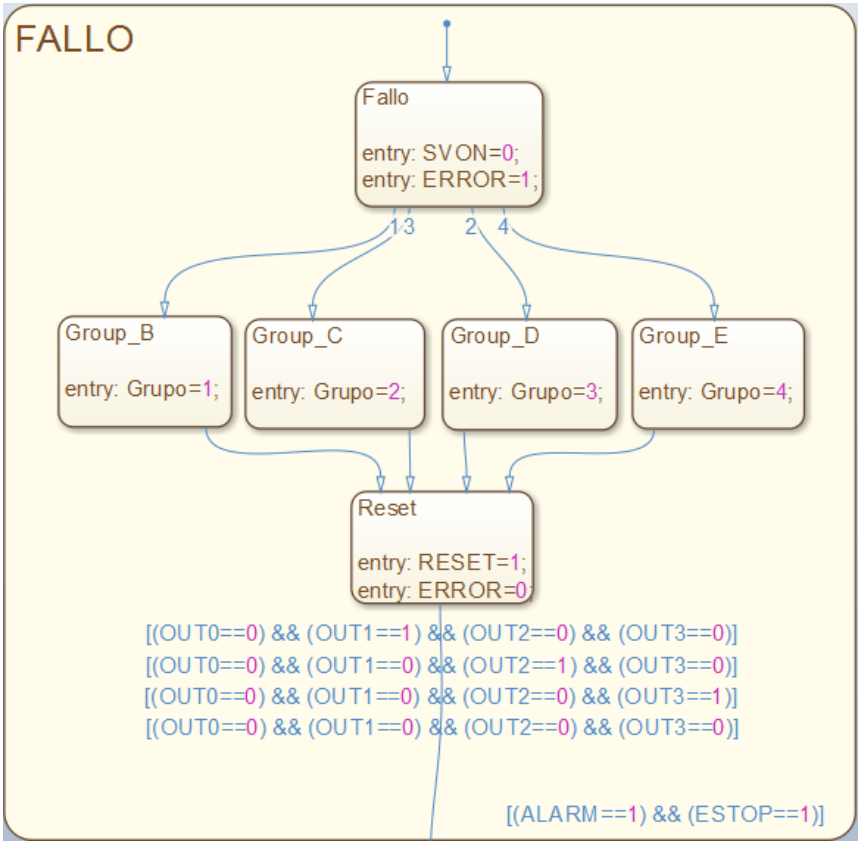
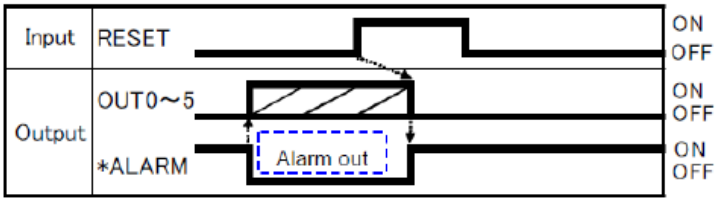


Figura 33: Modo Fallo



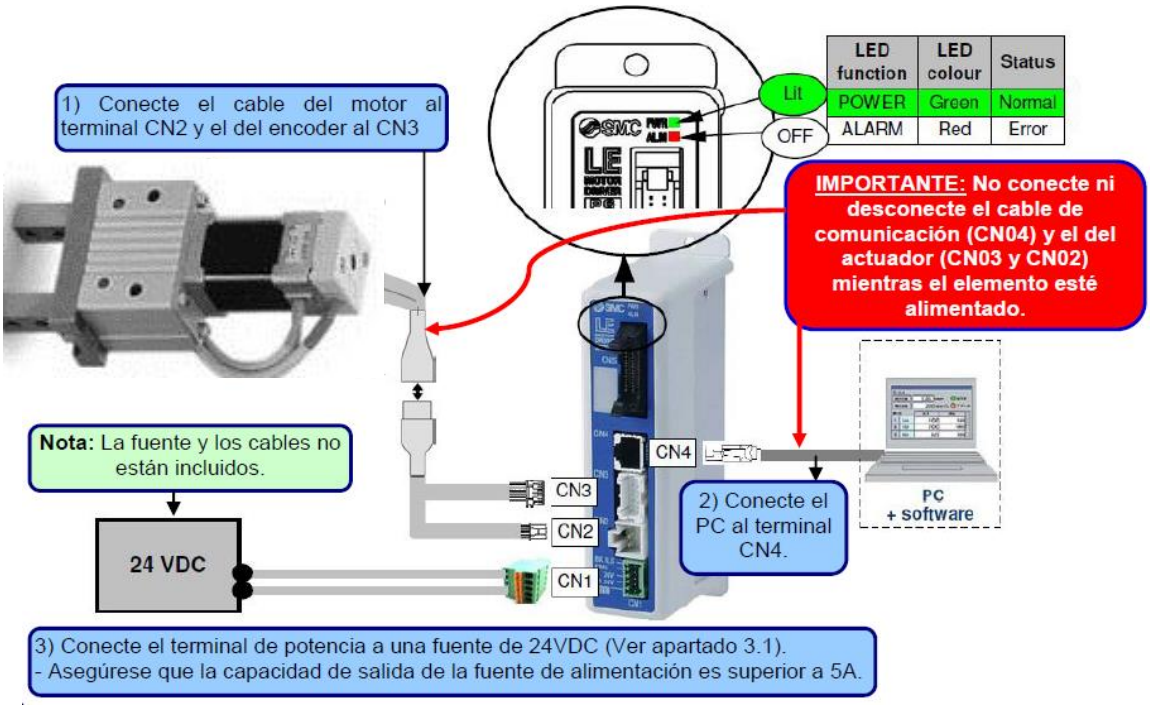
4.6 Programación del driver LEC-P6 mediante software

Antes de proceder a enviar órdenes desde *Arduino* al driver, este primero debe ser programado. La programación consiste en definir una serie de posiciones a las que la pinza debe dirigirse, ya sea en modo posicionamiento o en modo empuje. Dichas posiciones pueden verse como una estructura de datos que almacena parámetros de funcionamiento como la velicidad de movimiento, aceleraciones,…etc.

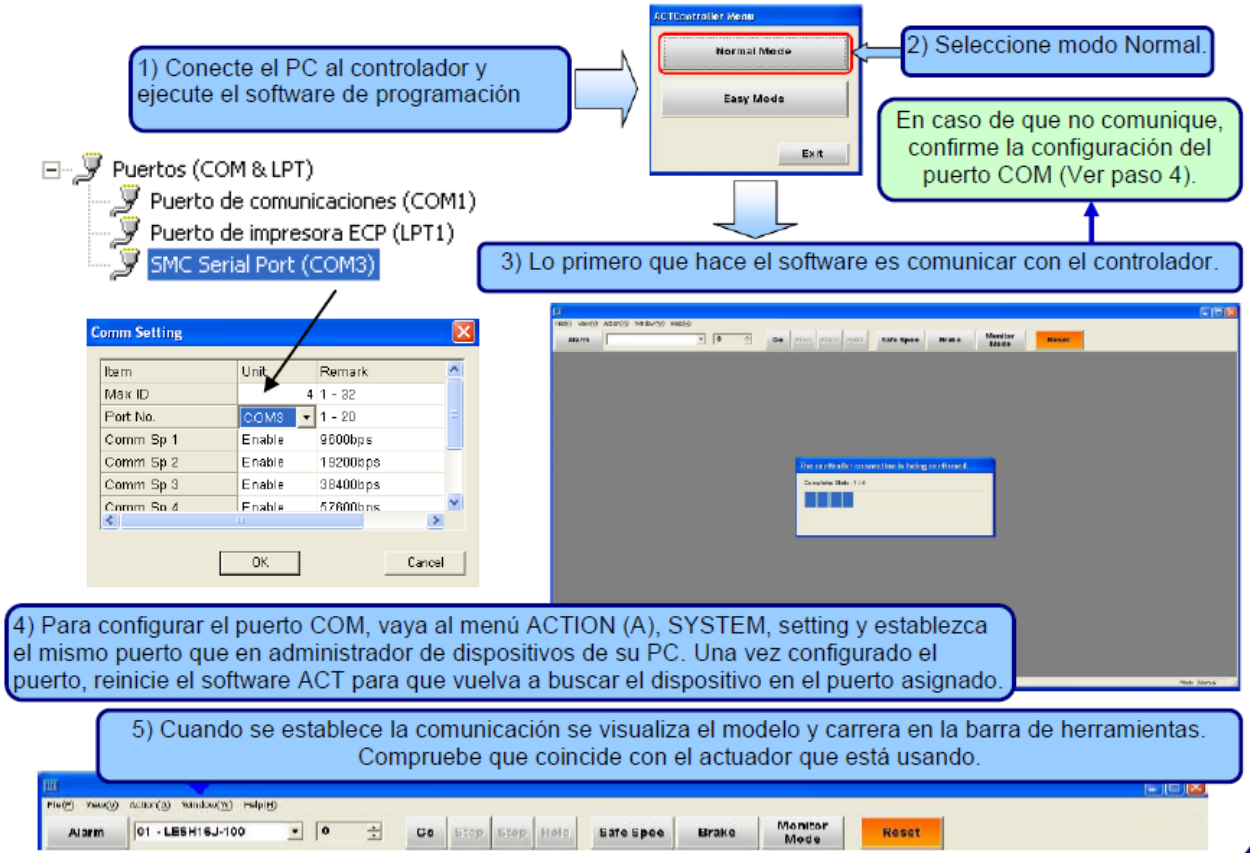
La programación del driver debe realizarse mediante el software *ACT Controller*, el cual se nos proporciona mediante un CD o bien podemos Descargar *ACT Controller* directamente desde la página web del fabricante.

A continuación, se enumeran los pasos que deben seguirse para una correcta programación del driver. Dichas instrucciones han sido obtenidas de la Guía Rapida de Puesta en Marcha del Controlador LEC-P6.

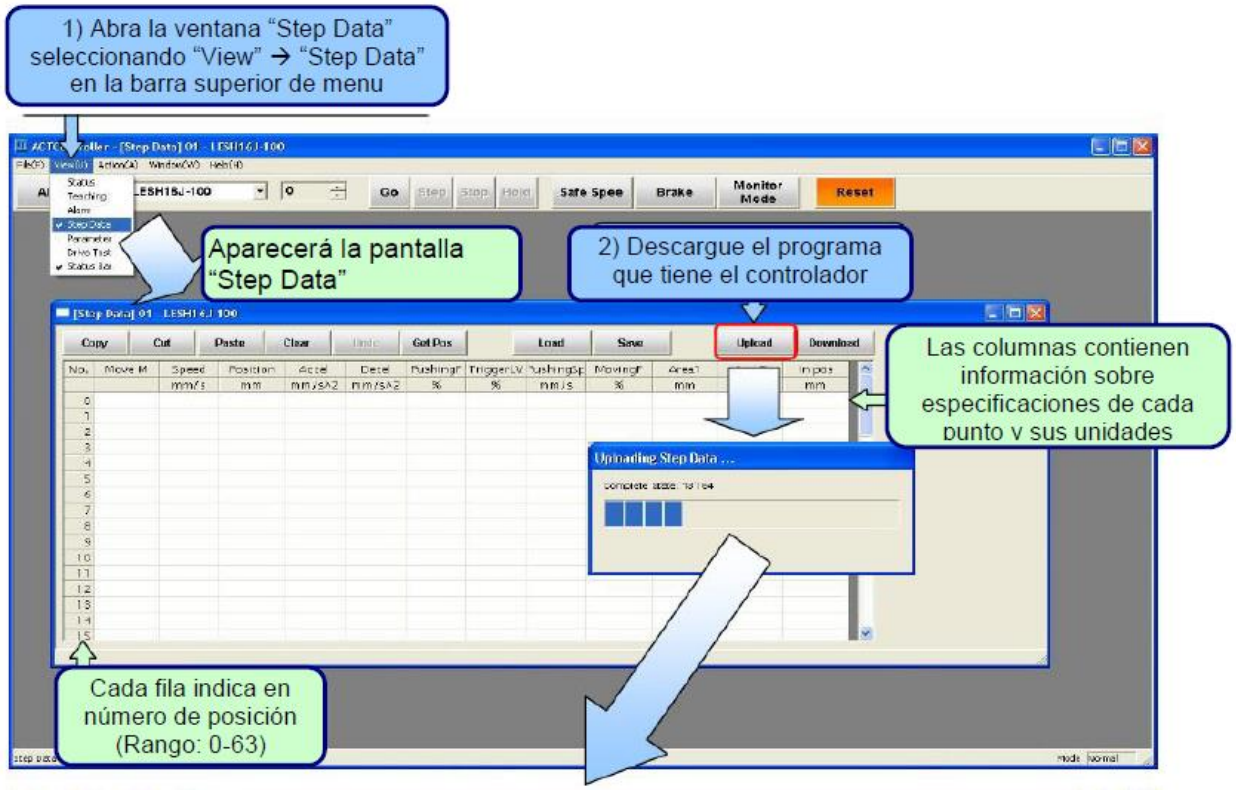
4.6.1 Conexión del driver al PC y a la pinza



4.6.2 Programación usando el modo normal del software



4.6.3 Operación de posicionado (Step Data)



No.	Move	Speed mm/s	Position mm	Accel mm/s²	Decel mm/s²	PushingF %	TriggerLV %	PushingSp mm/s	Moving F %	Area1 mm	Area2 mm	In pos mm
0	ABS	100	20.00	1000	1000	0	0	0	100	18.00	22.50	0.5
1	ABS	50	10.00	1000	1000	70	60	5	100	6.0	12.0	1.5

Máx. F durante el posicionado

Tipo de movimiento:
- ABS: Absoluto
- INC: Incremental

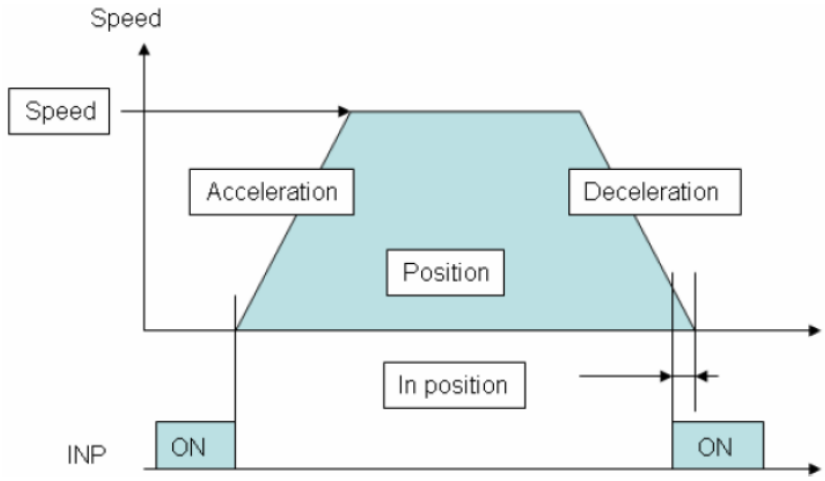
Si posición actual entre Area1 y Area2, la salida AREA = ON.

Si Area1 > Area2, la alarma 'Step Data ALM1' se activará.

Si Area1 = Area2, la salida AREA = OFF.

Modo posicionado: Cuando el actuador entra en el rango establecido +/- de la posición deseada, la salida INP = ON.

Nota: Los recuadros indicados en color rojo representan las columnas fundamentales de programación en modo posicionado (pantalla Step Data), para realizar un pick & place.



Este modo de operación se usa para programar la posición 1, es decir, la correspondiente a la orden PINZA_ABRIR ejecutada por *Arduino*. Cuando se manda dicha orden, la pinza se posiciona en su apertura máxima.

En la imagen correspondiente a nuestra programación en particular, podemos apreciar que la posición 1 corresponde a la operación de posicionamiento programada.

Para obtener más información sobre los campos programables de fuerza, velocidad…acudir al Anexo E: *Variables – Controlador LECP6 – Step Data*.

4.6.4 Operación de control de fuerza (Step Data)

No.	Move	Speed mm/s	Position mm	Accel mm/s^2	Decel mm/s^2	PushingF %	TriggerLV %	PushingSp mm/s	Moving F %	Area1 mm	Area2 mm	In pos mm
0	ABS	100	20.00	1000	1000	0	0	0	100	18.00	22.50	0.5
1	ABS	50	10.00	1000	1000	70	60	5	100	6.0	12.0	1.5

0 para modo 'posición'.

1 a 100 para modo 'fuerza'. El actuador se mueve a la posición especificada. A partir de dicha posición efectúa el empuje con el % de fuerza indicado en este campo.

La fuerza máxima depende del actuador.

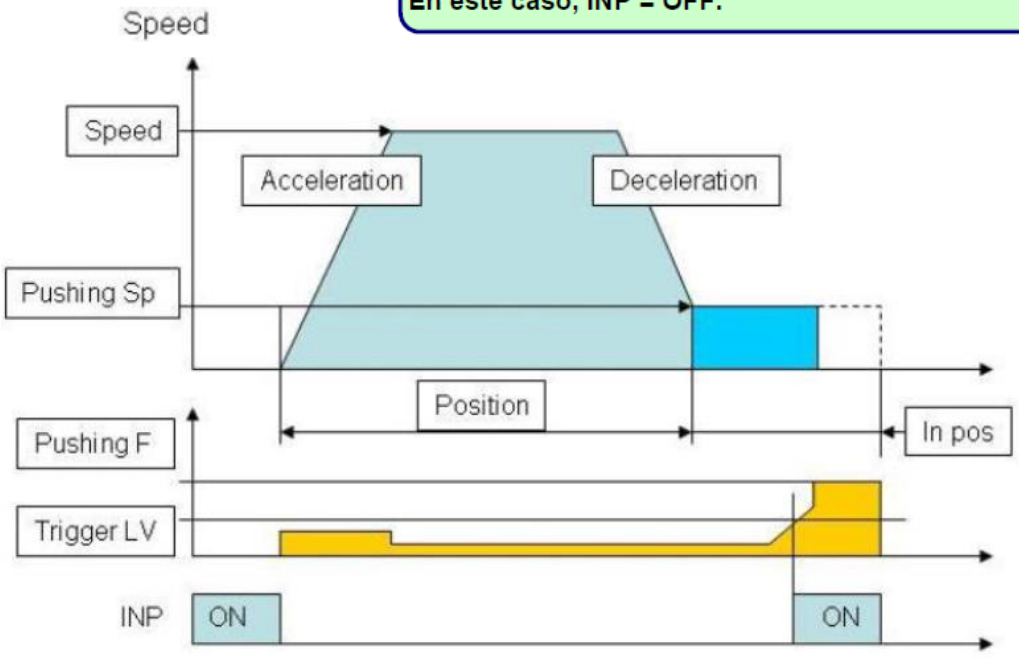
Sólo efectivo en modo 'fuerza'.

Define la velocidad de movimiento durante la operación de 'fuerza'.

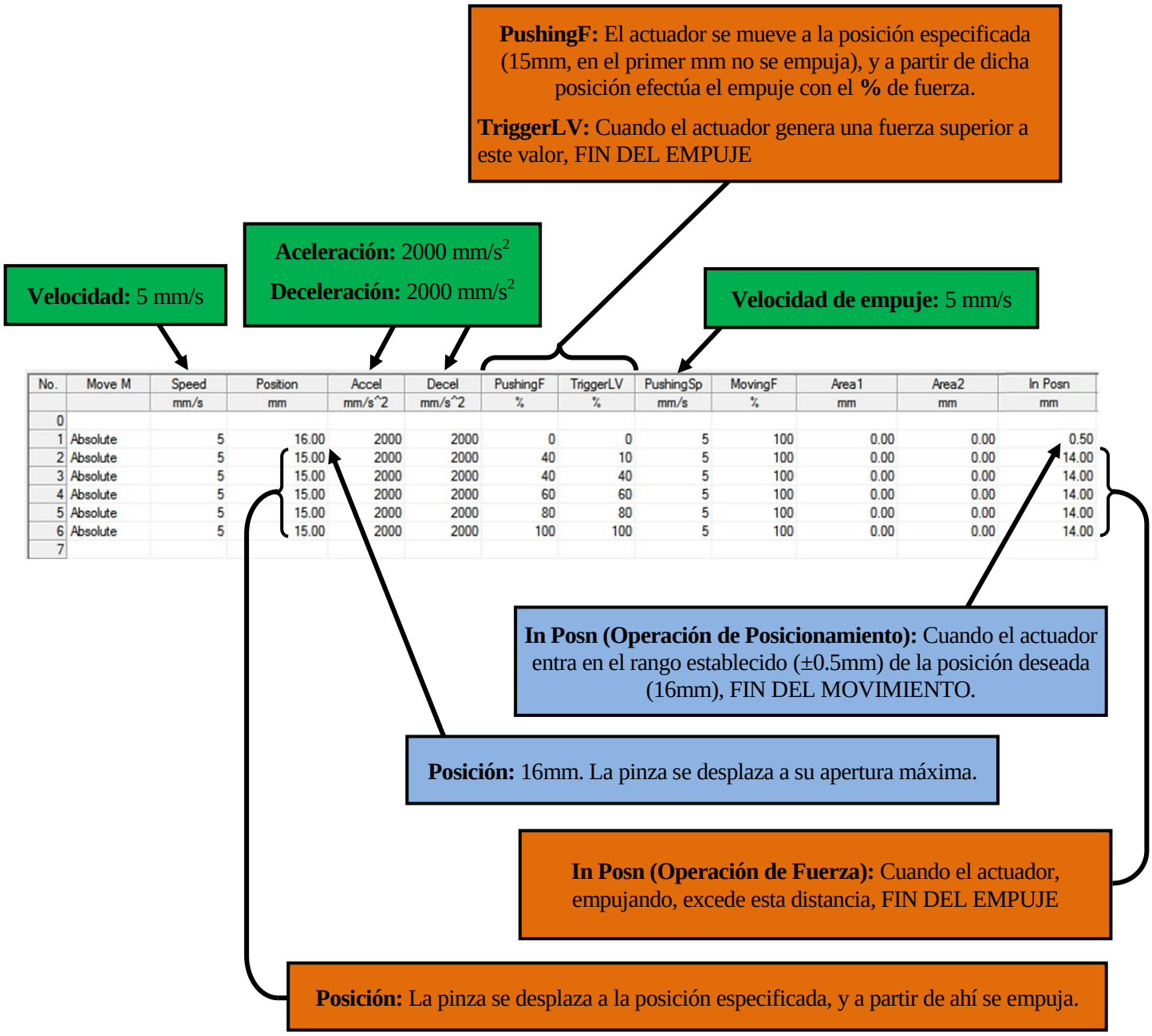
Sólo efectivo en modo 'fuerza'.

Cuando el actuador genera una fuerza superior a este valor, la salida INP = ON.

<u>Modo fuerza:</u> Cuando el actuador, empujando, excede esta distancia la operación de empujado se dará por finalizada. En este caso, INP = OFF.



Como se aprecia en la siguiente imagen, el campo numero 1 corresponde a la **operación de posicionamiento**, así como los campos del 2 al 6 corresponden a las 5 **operaciones de control de fuerza**. El resto son **parámetros generales.**



No.	Move M	Speed	Position	Accel	Decel	PushingF	TriggerLV	PushingSp	MovingF	Area1	Area2	In Posn
		mm/s	mm	mm/s^2	mm/s^2	%	%	mm/s	%	mm	mm	mm
0												
1	Absolute	5	16.00	2000	2000	0	0	5	100	0.00	0.00	0.50
2	Absolute	5	15.00	2000	2000	40	10	5	100	0.00	0.00	14.00
3	Absolute	5	15.00	2000	2000	40	40	5	100	0.00	0.00	14.00
4	Absolute	5	15.00	2000	2000	60	60	5	100	0.00	0.00	14.00
5	Absolute	5	15.00	2000	2000	80	80	5	100	0.00	0.00	14.00
6	Absolute	5	15.00	2000	2000	100	100	5	100	0.00	0.00	14.00
7												

5 DISEÑO CON CATIAV5 E IMPRESIÓN 3D

Una vez solucionada la compatibilidad electrónica entre los diversos sistemas, hay que resolver la adaptación física entre los mismos. Para ello, se han diseñado una serie de piezas que permiten la unión entre las diversas partes. Para ello, se ha hecho uso del software CAD/CAM/CAE *CatiaV5*.

CATIA es un programa informático de diseño, fabricación e ingeniería asistida por computadora comercial realizado por Dassault Systèmes. El programa está desarrollado para proporcionar apoyo desde la concepción del diseño hasta la producción y el análisis de productos.

5.1 Piezas modeladas en CatiaV5

5.1.1 Pinza

La empresa SMC Corporation nos ofrece en su página web los planos 2D de la pinza *LEHZ25K2-14*. Dichos planos, cuentan con todas las cotas necesarias para modelar en 3D la geometría completa del elemento.

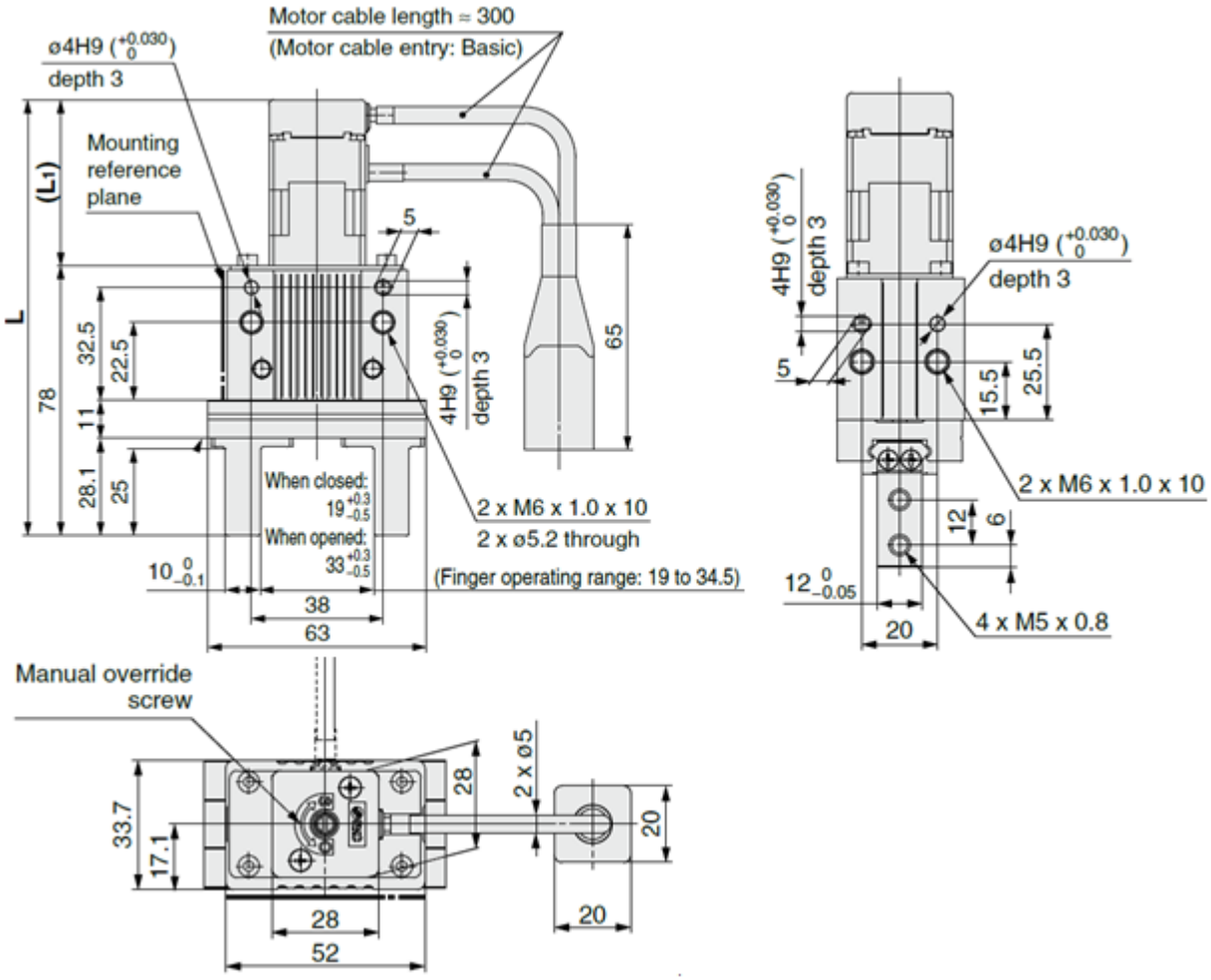


Figura 34: Alzado, planta y perfil de la pinza *LEHZ25K2-14*

A partir de dichos planos, se genera la pieza con la mayor exactitud posible.

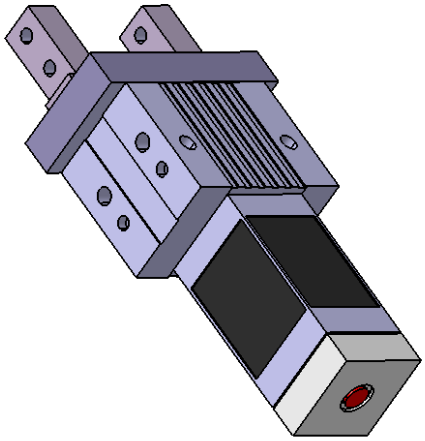


Figura 35: Pinza *LEHZ25K2-14* modelada en CatiaV5

5.1.2 Muñeca del robot

Esta pieza no ha sido modelada en Catia por el autor. La empresa ABB nos suministra en su página web los modelos CAD de todos sus productos en distintos formatos.

Enlace: Modelos CAD del IRB120

Modelos CAD del IRB 120

Ver.	DXF/DWG 2D	SAT	SW	STEP	PARASOLID	VDA	STL	RobotStudio	Catia V5
3-58	joint	joint	comp	joint	joint	joint	sim	sim	joint

Nos descargamos los planos en formato CatiaV5 y abrimos el archivo que contiene al *LINK6*, es decir, la ultima extremidad del robot donde se acoplan las herramientas.

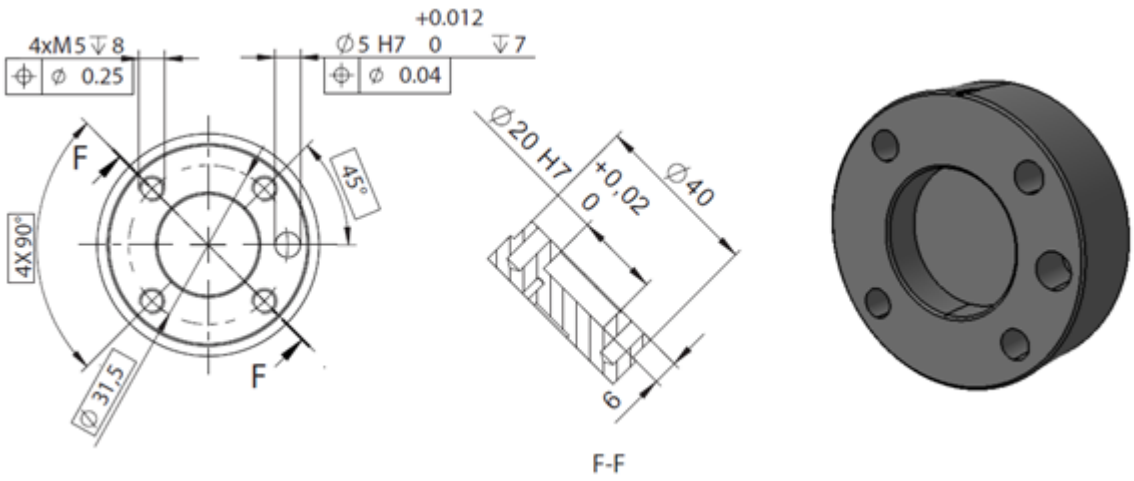


Figura 36: Muñeca del Robot IRB120

5.1.3 Pieza de unión

Una vez modelada la geometría 3D de ambas partes a unir, diseñamos una pieza cuyos extremos se adapten a las dimensiones y taladros específicos de cada pieza. El resultado es el siguiente:

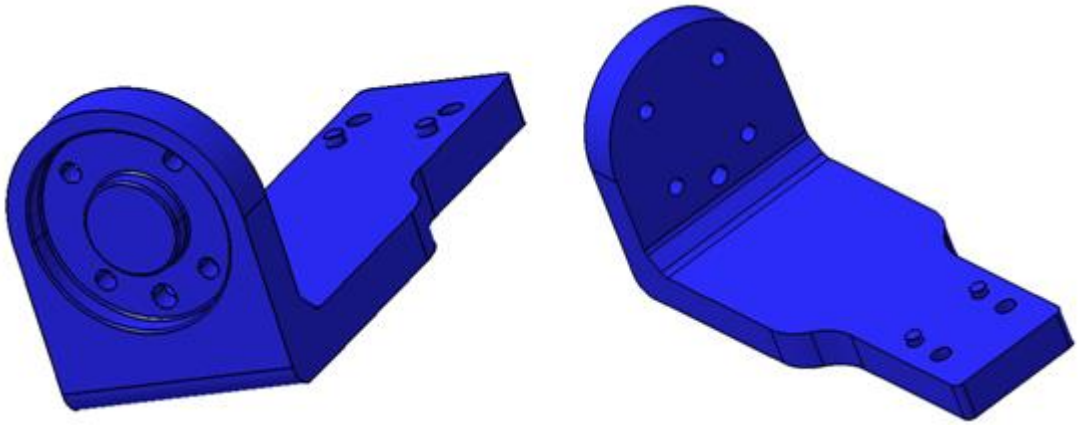


Figura 37: Pieza de unión modelada en CatiaV5

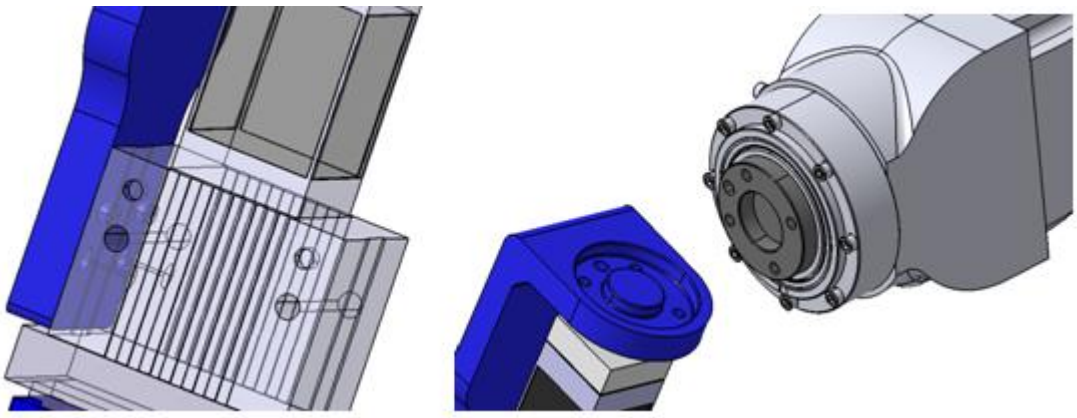


Figura 38: Unión entre pinza, pieza y muñeca del robot

5.1.4 Dedo de la pinza

La pinza cuenta con dos dedos de 25mm. Para poder manipular objetos más grandes, se han diseñado y fabricados dos fundas que se acoplan a la pinza y hacen las veces de garras.

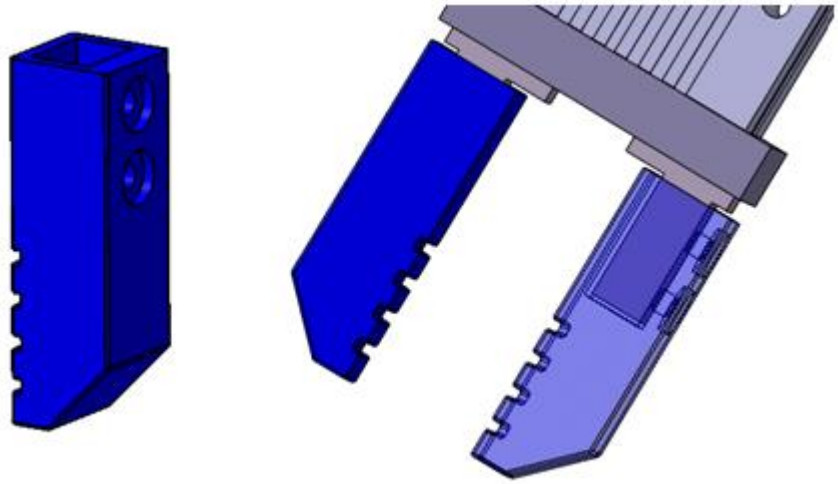


Figura 39: Dedo de la pinza

5.1.5 Mecanismos para RobotStudio

A la hora de modelar las piezas, tanto la pinza, como los dedos y la pieza de unión se han diseñado como *CATParts* individuales. Como veremos más adelante, el programa de RobotStudio requiere geometrías monobloque para crear herramientas y mecanismos. Es por ello que se han creado diversos *CATProducts* (uniones de varios *CATParts*) para crear dichas geometrías.

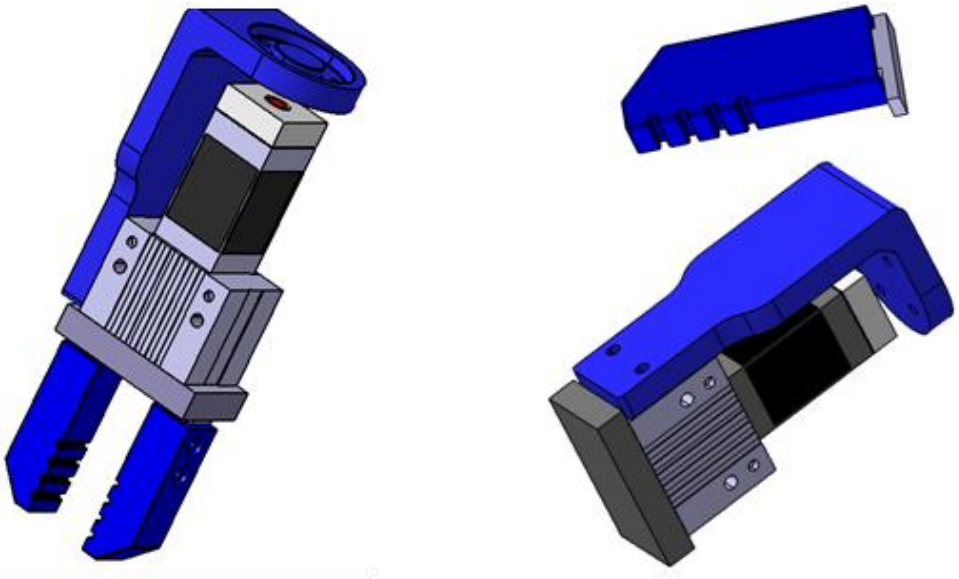


Figura 40: Pinza completa, y mecanismo formado por la base y los dedos

5.1.6 Soporte para interruptor

El interruptor bipolar requiere un encapsulado para poder fijarlo al carril DIN normalizado que soporta todos los componentes. Para ello se ha optado por diseñarlo y fabricarlo.

5.1.6.1 Encapsulado

A partir de las dimensiones del interruptor, se diseña un recubrimiento que permita la salida trasera de los cables de conexión.

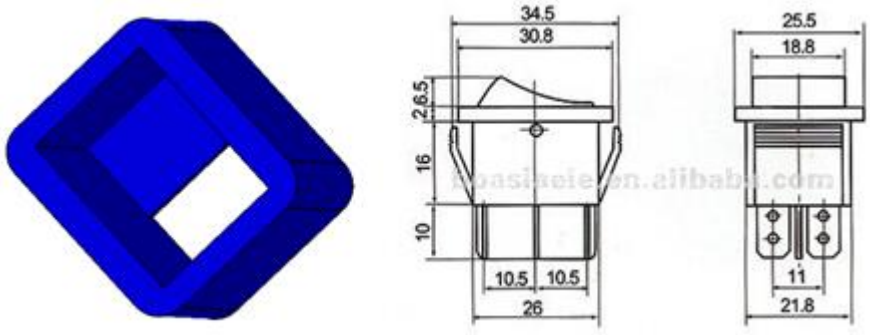


Figura 41: Encapsulado del interruptor

5.1.6.2 Sujeción a carril DIN

Para poder anclar el encapsulado al carril DIN, debemos conocer las dimensiones de la pieza de unión que se ancla a dicho carril. Las medidas que nos interesan son los 36mm de separación entre ejes y el diámetro de los mismos.

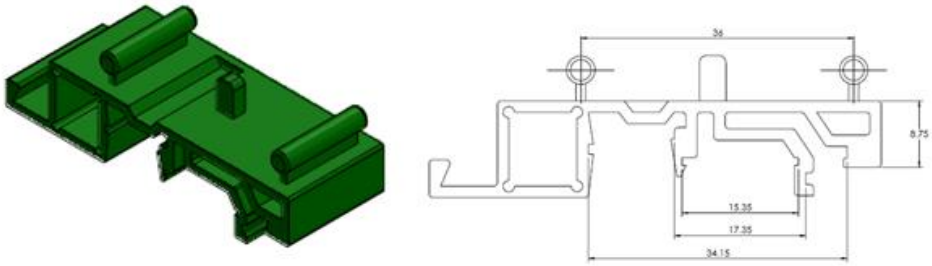


Figura 42: Pieza comercial que une el carril DIN con el soporte

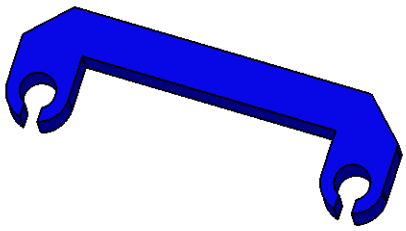


Figura 43: Pieza fabricada

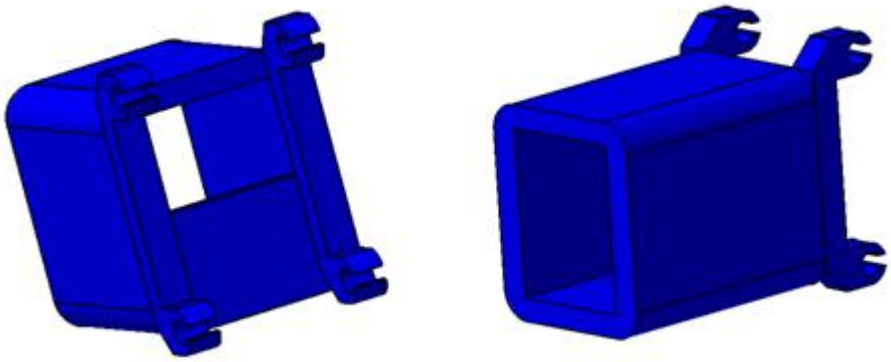


Figura 44: Montaje final

5.1.7 Caja PCB

La PCB también va a ser montada sobre el carril DIN que llevará todos los componentes. Para poder alojarla, se ha diseñado una protección que la sostenga y la recubra, y que además permita la libre circulación del aire para evitar un calentamiento excesivo de los componentes electrónicos.

Además también se han adosado las mismas sujeciones al carril DIN antes definidas.

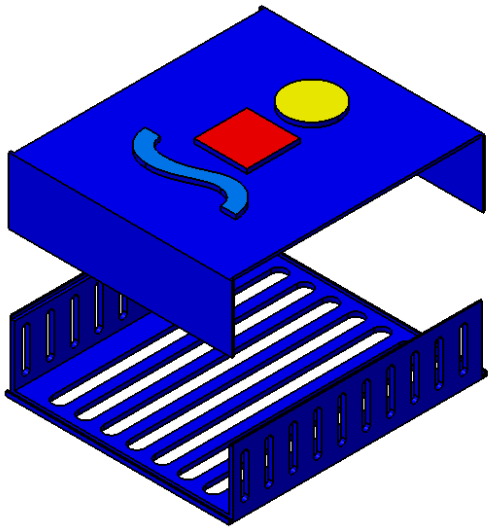
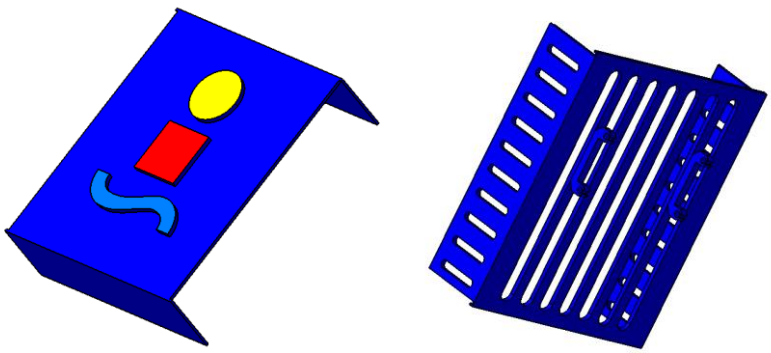


Figura 45: Caja de la PCB formada por las partes superior e inferior

5.1.7.1 Parte superior e inferior



5.1.8 Anillas de sujeción de cables

Con el fin de que no se produzcan enredos entre el robot y el cable de la pinza, se han diseñado unas piezas que alejan y canalizan el recorrido de dicho conector. Los taladros de dichas piezas están en consonancia con las especificaciones de los orificios para montajes de equipos adicionales.

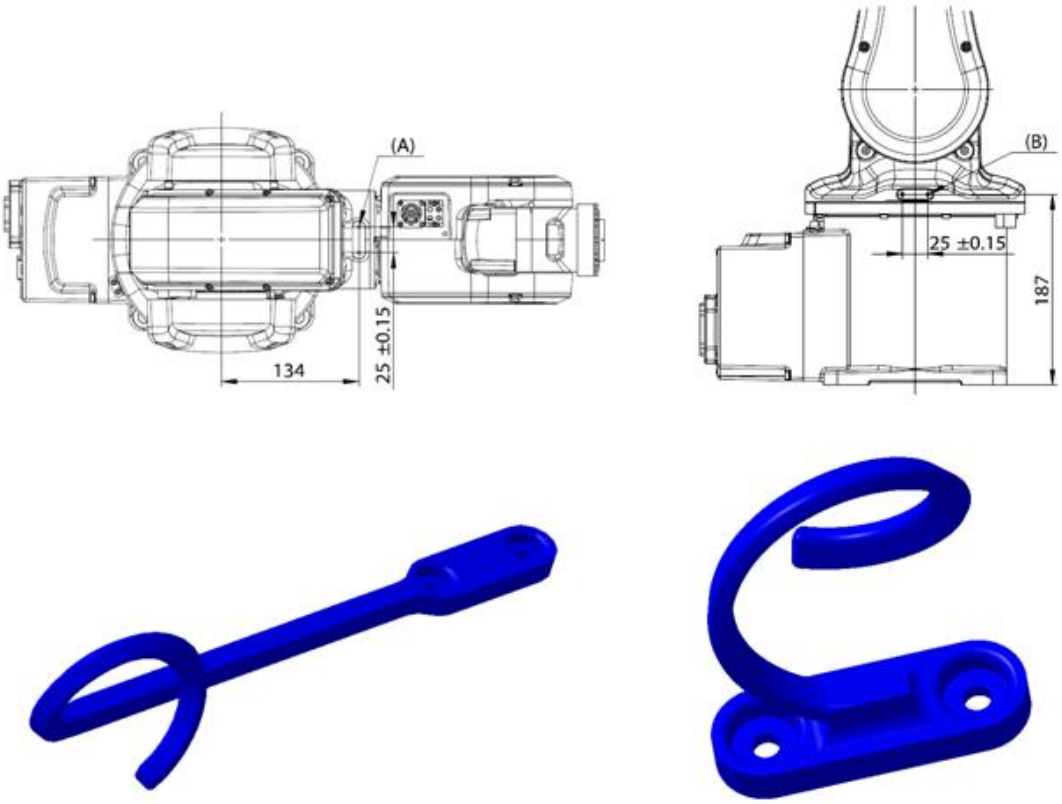


Figura 46: Anillas para cables

5.1.9 Mesa de trabajo

Como más adelante se detalla en el capítulo dedicado a *RobotStudio*, es necesario disponer de las geometrías que forman el entorno de trabajo del robot para crear las estaciones virtuales. Es por ello que se han tomado medidas reales de las mesas de trabajo que dispone el Laboratorio de Robótica de la Escuela de Ingeniería de Sevilla.

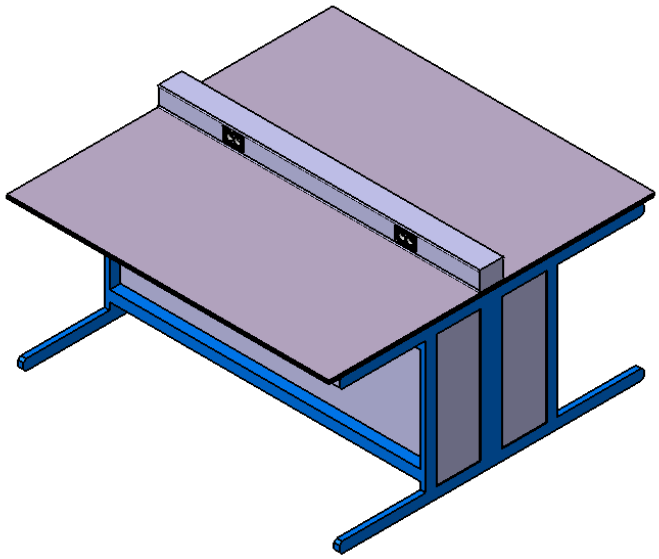


Figura 47: Mesa del laboratorio

5.1.10 Laboratorio

También se ha modelado el Laboratorio al completo. Las partes que no están cercanas al entorno de trabajo de robot, como la pizarra o las paredes, no alcanzan un nivel de detalle tan alto como la mesa, la cual si es muy semejante a la real.

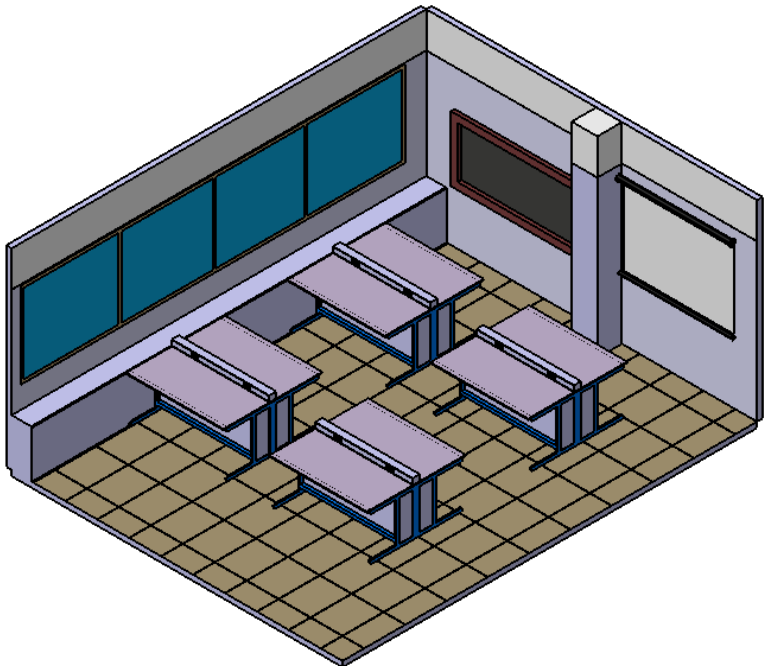


Figura 48: Laboratorio de robótica

5.2 Piezas fabricadas en 3D

5.2.1 Fabricación aditiva

La impresión 3D es un grupo de tecnologías de fabricación por adición donde un objeto tridimensional es creado mediante la superposición de capas sucesivas de material. Las impresoras 3D son por lo general más rápidas, baratas y fáciles de usar que otras tecnologías de fabricación por aditiva.

5.2.2 Impresora 3D Prusa i3 hephestos

La impresora 3D Prusa i3 Hephestos es un proyecto libre diseñado y desarrollado por el departamento de Innovación y Robótica de BQ.

Área de impresión	X	215 mm	
	Y	210 mm	
	Z	180 mm	
Resolución		100 micras	
Velocidad		50 mm/s	
Material		PLA 1.75 mm	

Tabla 15: Características de la impresora 3D Prusa i3 hephestos

5.2.3 Piezas fabricadas

A continuación se expone el resultado algunas piezas obtenidas mediante fabricación aditiva.

Figura 49: Piezas 3D

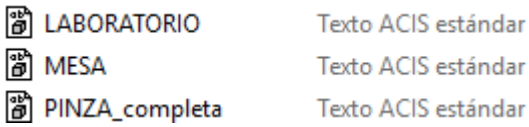
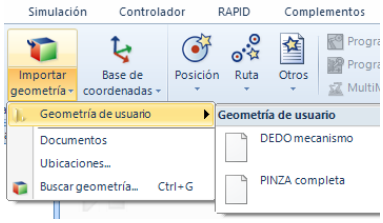
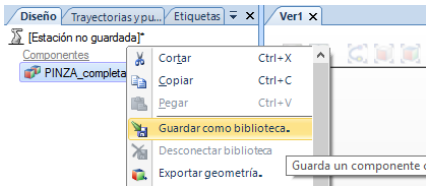
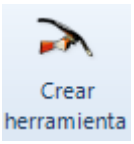
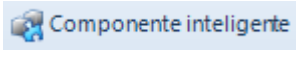
6 ROBOTSTUDIO

6.1 Introducción

RobotStudio es una aplicación de PC destinada al modelado, la programación fuera de línea y la simulación de células de robot. Le permite trabajar con un controlador fuera de línea, que constituye un controlador IRC5 virtual que se ejecuta localmente en su PC, y también con un controlador IRC5 físico real.

A la hora de trabajar en este entorno, utilizaremos una **estación virtual** que reproduzca las condiciones físicas, geométricas y mecánicas de una forma realista y cercana a la realidad. Así, seguiremos una serie de pasos para crear nuestra estación virtual, así como para definir los elementos que posteriormente se utilizaran en el laboratorio.

La secuencia de pasos para crear nuestra estación virtual será la siguiente:

1	**Geometrías modeladas con Catia V5**	
2	**Conversión mediante 3D Tool V12**	
3	**Añadir a la carpeta Documentos, RobotStudio, Geometry**	
4	**Guardar como Biblioteca (Desde RobotStudio)**	
5	**Creación de una herramienta**	
6	**Creación de un mecanismo**	
7	**Creación de un Smart Component**	

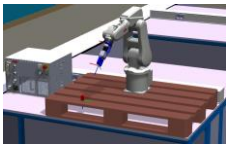
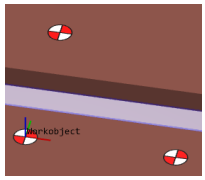
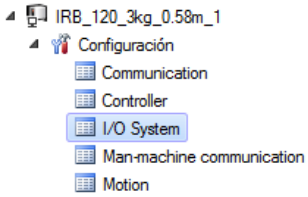
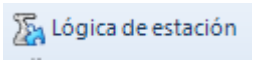
8	**Creación de la estación virtual: colocación de las geometrías**	
9	**Creación del Objeto de trabajo**	
10	**Creación de entradas y salidas**	
11	**Creación de la lógica de estación**	

Tabla 16: Secuencia de pasos para crear estación virtual

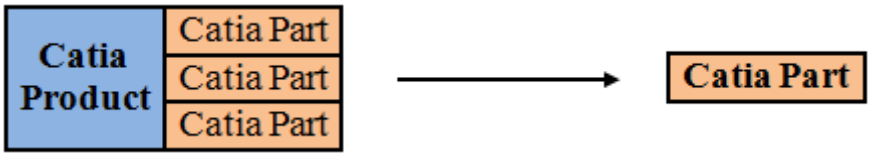
6.2 Creación de las geometrías de la estación

Para crear herramientas, mecanismos y componentes inteligentes, primero debemos partir de una geometría. Una geometría se compone de datos de CAD que han sido creados previamente mediante algun software de diseño 3D, y que puede ser importada para su uso en *RobotStudio*. La extensión de los archivos CAD utilizados que es compatible con *RobotStudio* es la **.sat**

Para convertir un archivo generado en CatiaV5 a un archivo **.sat** hay que seguir una serie de pasos.

6.2.1 Modelado del CatPart

Cuando creamos un *Part* desde *CatiaV5*, lo que obtenemos es un archivo *CATPart* que podemos convertir directamente a **.sat**. En el caso de que hayamos realizado un *CATProduct*, debemos convertir dicho *Product* a un solo *Part*. Para ello, desde el *Assembly Desing*, debemos pulsar en *Tools* y después en *Generate CATPart from Product.*



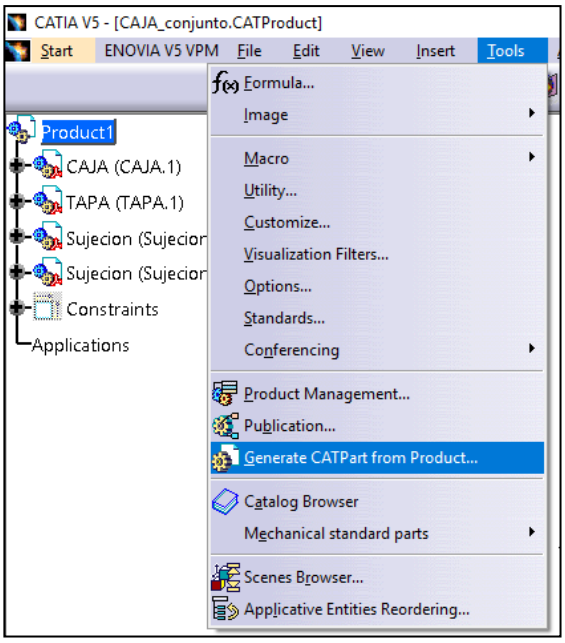


Figura 50: Convertir Product en Part

6.2.2 Conversión a .sat

Para convertir a **.sat** debemos usar algún programa de conversión de archivos CAD. En este caso, se ha utilizado el programa *3D Tool V12*. Para convertir un archivo, debemos abrir el programa. En el campo *File to convert* hay que insertar el archivo CATPart a convertir, y en *Output file* especificamos en directorio de generación, así como el formato de salida, que debe ser *ACIS SAT*.

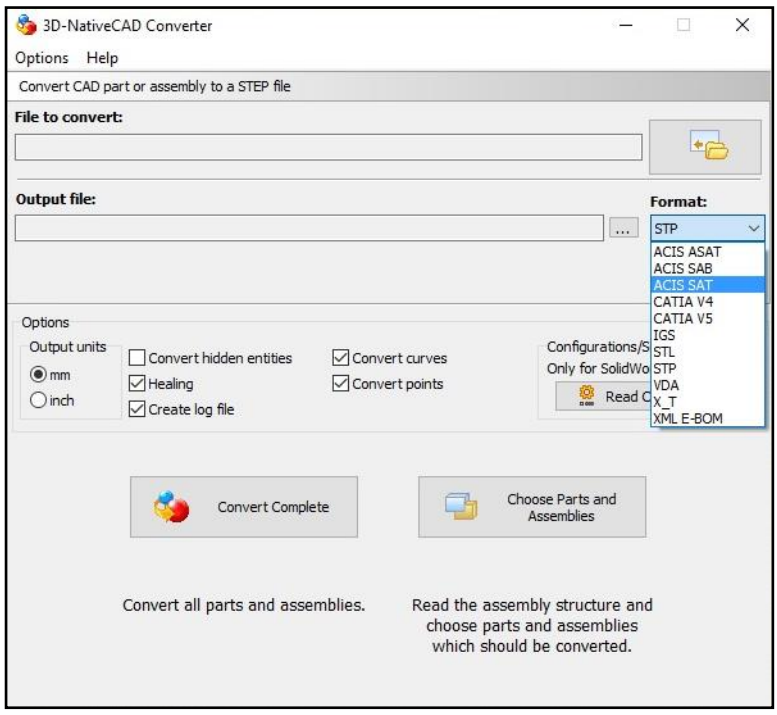


Figura 51: 3D Tool V12

6.2.3 Añadir a la geometría de usuario

Una vez tengamos los archivos **.sat**, solo hay que copiarlos y pegarlos en *Documentos-RobotStudio-Geometry*. De ese modo, al abrir RobotStudio y buscar en *Importar Geometría-Geometría de Usuario*, aparecerán nuestros modelos con las diversas extensiones en las que los hayamos incorporado.

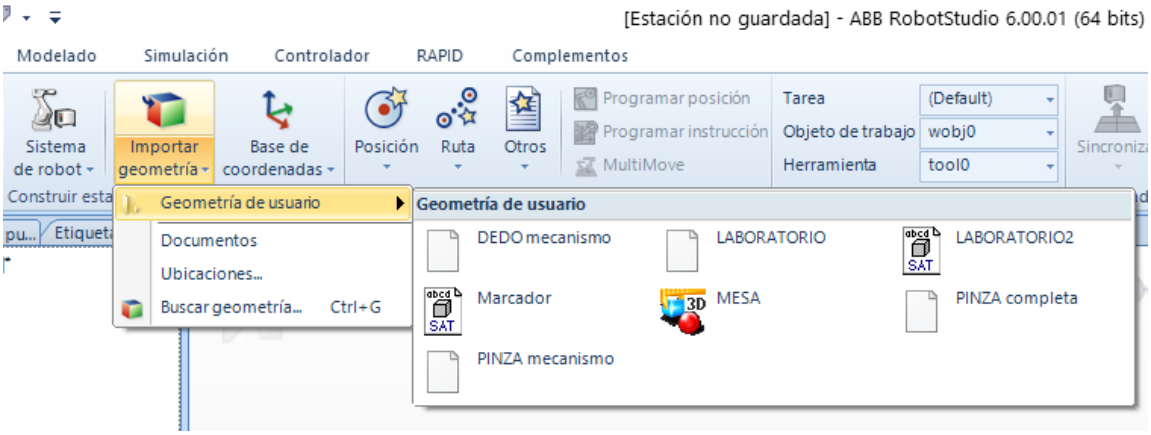


Figura 52: Geometría de usuario

6.2.4 Añadir a la biblioteca de usuario

Con la pieza cargada en nuestra pestaña de *Diseño*, podemos hacer click derecho y añadirla como un componente de biblioteca. Al hacer esto, automáticamente nuestra pieza pasa también a la carpeta *Documentos-RobotStudio-Libraries* y ya la tendremos siempre disponible desde Robotstudio accediendo a *Importar Biblioteca, Biblioteca de usuario.*

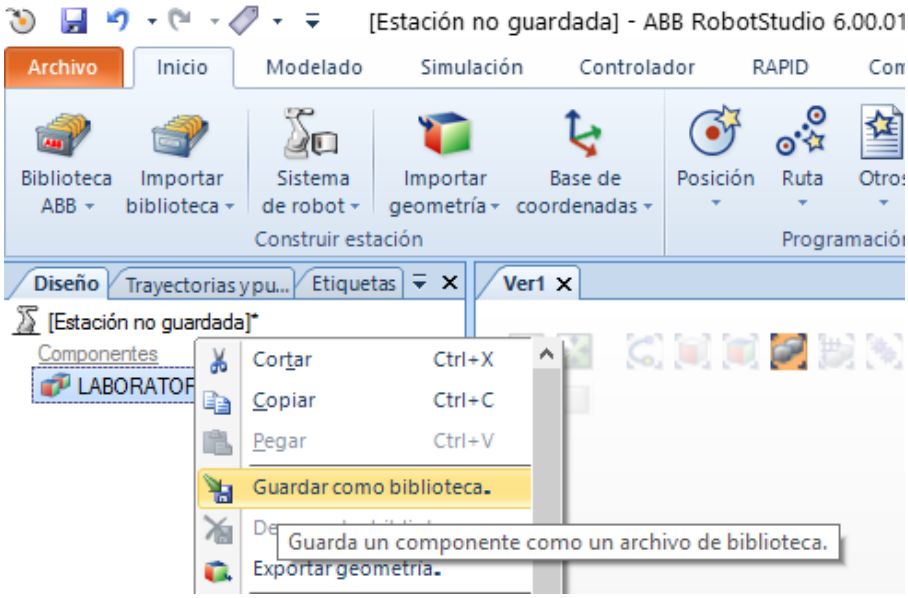


Figura 53: Guardar como biblioteca

Figura 54: Biblioteca de usuario

	Geometría	**Librería**
Archivo guardado en…	*Documentos-RobotStudio-Geometry*	*Documentos-RobotStudio-Libraries*
Se accede desde…	*Importar Geometría, Geometría de Usuario*	*Importar Biblioteca, Biblioteca de usuario*
Formato	MESA Texto ACIS estándar PINZA_completa Texto ACIS estándar PINZA_mecanismo Texto ACIS estándar	MESA Archivo de biblioteca de RobotStudio PINZA Archivo de biblioteca de RobotStudio PINZA_mecanismo Archivo de biblioteca de RobotStudio

Figura 55: Diferencias entre geometría y librería

6.3 Creación de una herramienta

Para simular la pinza del robot, necesita los datos de herramienta. Dichos datos contienen la información necesaria para mover y trabajar con la herramienta.

6.3.1 Crear herramienta

Hacer clic en *Crear herramienta* y seleccionar *Usar existente* y la herramienta importada de la lista.

6.3.2 Propiedades

A continuación, introducir la *Masa* de la herramienta, su *Centro de gravedad* y el *Momento de inercia Ix, Iy, Iz*, los cuales pueden obtenerse mediante *CatiaV5*.

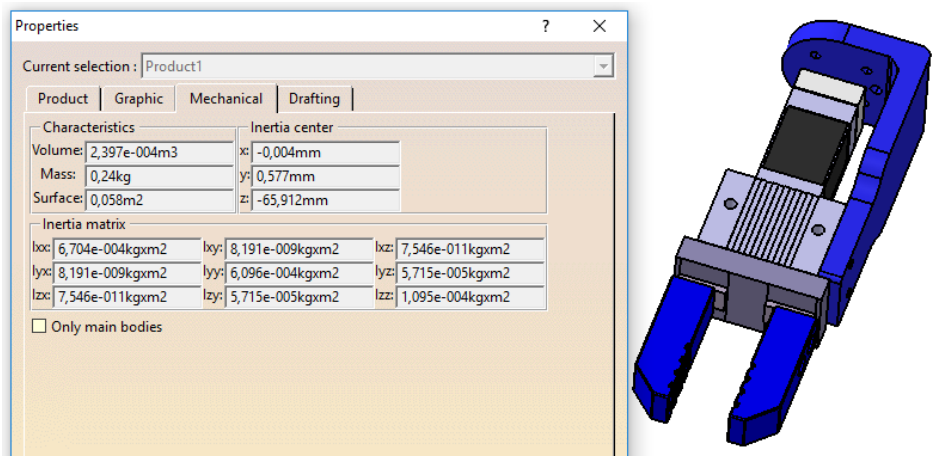


Figura 56: Propiedades de la herramienta

6.3.3 TCP

Introducir un nombre para el punto central de la herramienta (TCP). El TCP de la herramienta pinza tiene la misma orientación que la muñeca del robot, por lo que no hay que modificar este campo. Solo hay que editar la distancia en Z desde el extremo de la muñeca hasta la punta de los dedos. Dicho dato se obtiene nuevamente midiendo en Catia.

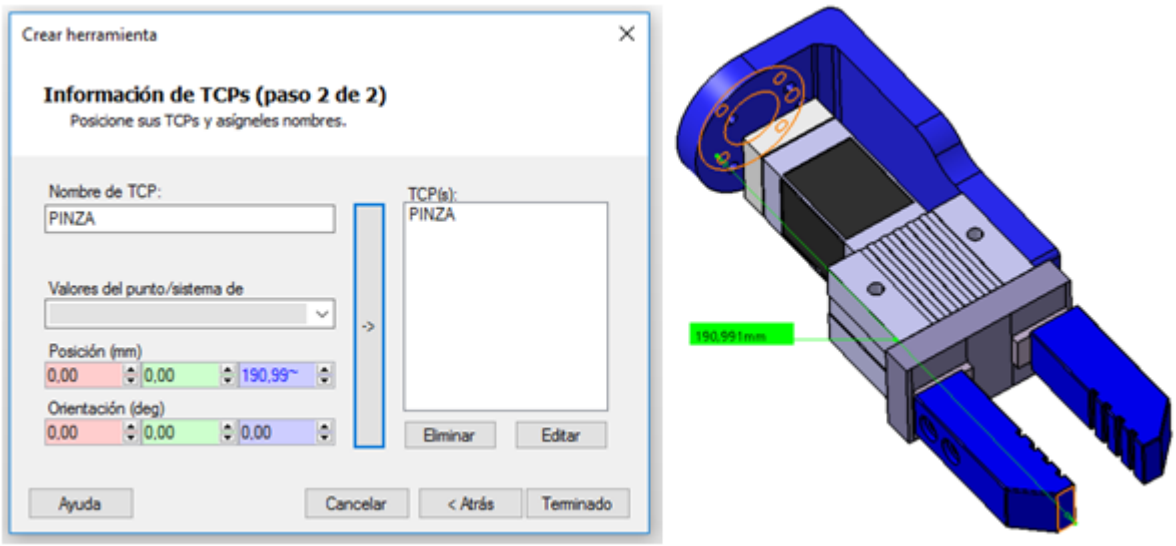


Figura 57: TCP

El procedimiento nos genera una herramienta cuyos ejes coordenados de base coinciden con los ejes de la articulación de la muñeca, y cuyo TCP se sitúa en el extremo centro de los dedos de agarre.

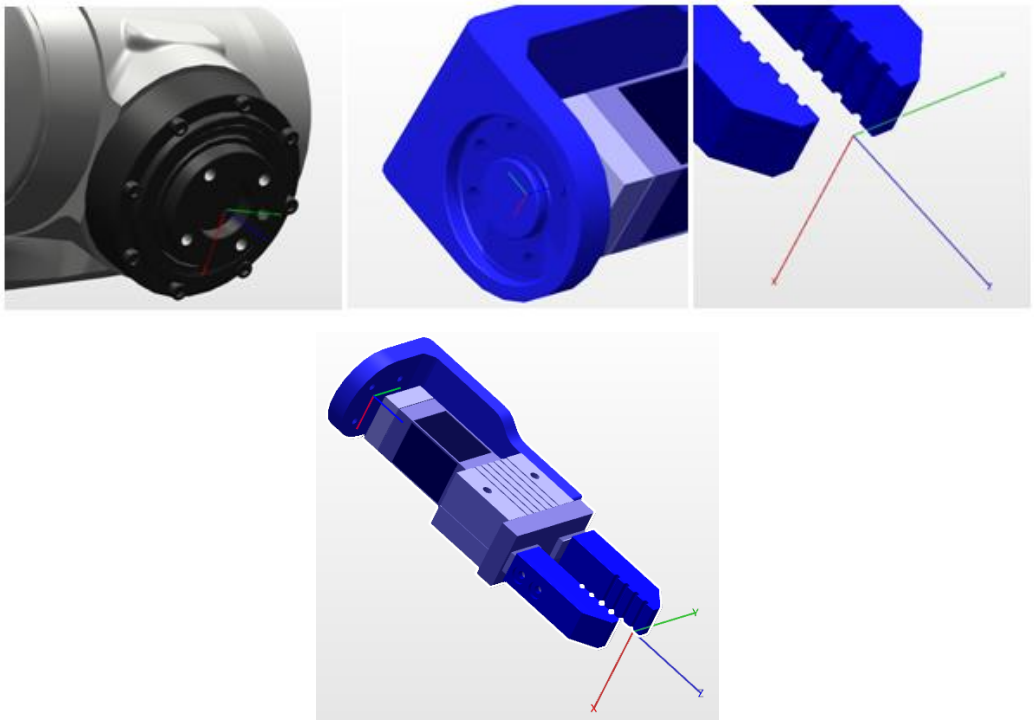


Figura 58: Herramienta PINZA

6.4 Creación de un mecanismo

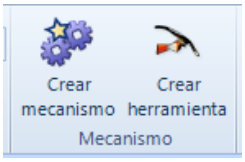
Un mecanismo es una representación grafica de un robot, eje externo, herramienta o dispositivo. Las distintas partes de un mecanismo se mueven a lo largo de ejes o alrededor de ellos. La creación de un mecanismo depende de la construcción meticulosa de los nodos principales de la estructura de árbol. Cuatro de ellos (eslabones, ejes, bases de coordenadas/herramientas y calibración) aparecen marcados en rojo inicialmente.

A medida que cada nodo sea configurado con suficientes nodos secundarios para que sea válido, el color cambia al verde. Tan pronto como todos los nodos son válidos, el mecanismo se considera compilable y puede ser creado.

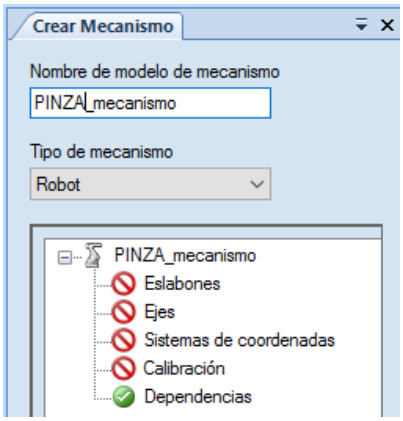
Nodo	Criterios de validez
Eslabones	• Contiene más de un nodo secundario. • Está definido el eslabón base. • Todas las partes del eslabón permanecen en la estación.
Ejes	• Al menos una articulación debe estar activa y debe ser válida.
Datos de bases de coordenadas/herramientas	• Debe existir al menos un dato de base de coordenadas/herramienta. • En el caso de un dispositivo, no se requiere ninguna base de coordenadas.
Calibración	• En el caso de un robot, se requiere exactamente una calibración. • En el caso de un eje externo, se requiere una calibración para cada articulación. • En el caso de una herramienta o un dispositivo, las calibraciones se aceptan, pero no son obligatorias.

6.4.1 Crear mecanismo tipo herramienta

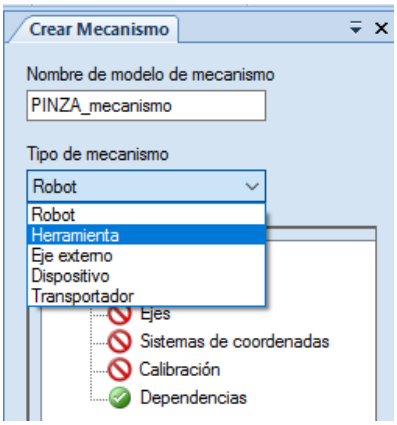
- El primer paso es hacer click, dentro de la pestaña Modelado, en **Crear mecanismo**. Se abre el modelador de mecanismos en el modo de creación.



- En el cuadro **Nombre de modelo de mecanismo**, introducir un nombre de mecanismo. En nuestro caso, haremos el PINZA_mecanismo.

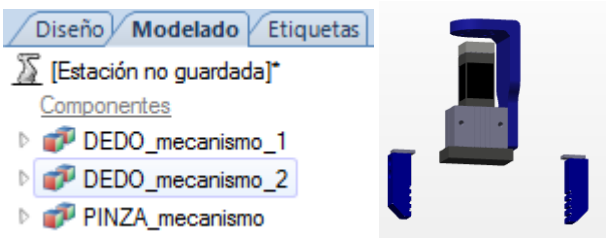


- En la lista **Tipo de mecanismo**, seleccionar un tipo de mecanismo. En nuestro caso, herramienta, dado que la pinza va a usarse como la herramienta del robot.

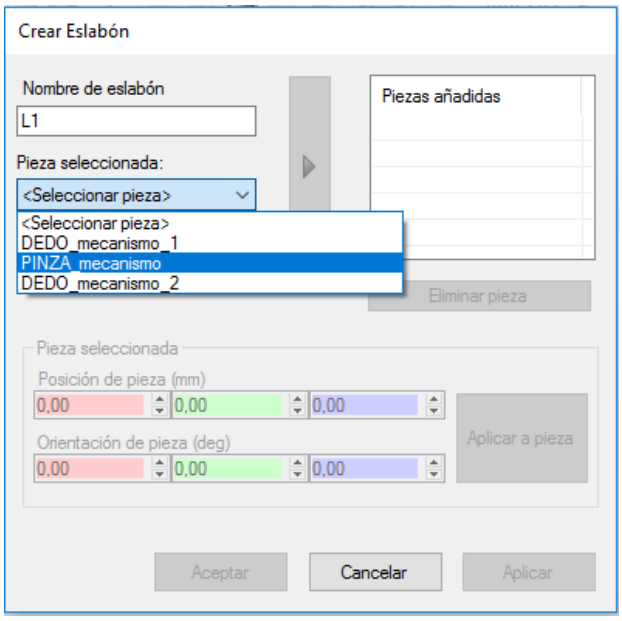


6.4.2 Eslabones

- En la estructura de árbol, hacer clic con el botón derecho en **Eslabones** y a continuación clic en **Añadir eslabón.** Para que nos aparezca algún eslabón para añadir, previamente debemos haber importado las geometrías de usuario que conforman nuestro mecanismo. En nuestro caso la base y dos dedos. **Nota:** No es posible seleccionar las piezas que formen parte de una biblioteca o un mecanismo.

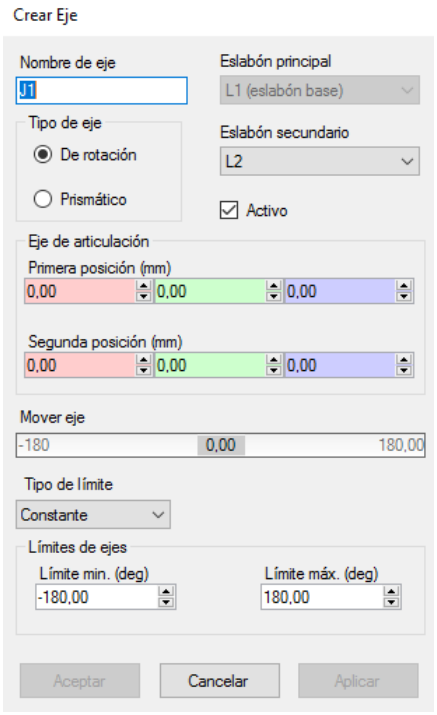


- **Crear eslabón**. En **Nombre de eslabón** aparece una propuesta de nombre. En la lista **Pieza seleccionada**, seleccionar una pieza (que se resaltará en la ventana de gráficos) y hacer clic en el botón de flecha para añadir la pieza al cuadro de lista Piezas. Seleccionar una pieza del cuadro de lista Piezas, como por ejemplo, el eslabon base llamado PINZA_mecanismo. Introducir valores en los cuadros de grupo **Pieza seleccionada** y a continuación hacer clic en **Aplicar a pieza**. Repetir el proceso con cada pieza según sea necesario colocando debidamente los eslabones según las dimensiones reales de nuestra pinza. Haga clic en **Aceptar**.



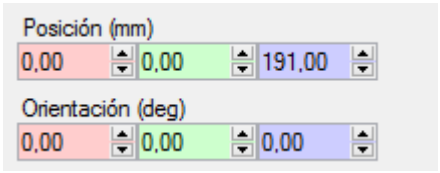
6.4.3 Ejes

- En la estructura de árbol, hacer clic con el botón derecho en **Ejes** y a continuación clic en **Añadir eje** para mostrar la ventana de diálogo **Crear eje**. En **Nombre de eje** aparece una propuesta de nombre. Completar la ventana de diálogo **Crear eje** y a continuación clic en **Aceptar**. Debemos establecer el movimiento máximo y mínimo del eje. En nuestro caso, dado que la apertura máxima de la pinza es de 16mm, cada uno de estos ejes debe moverse como máximo 8mm.



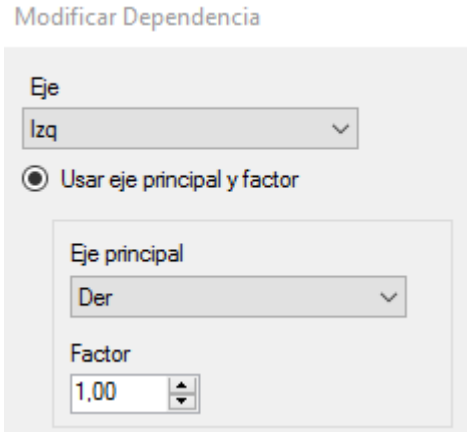
6.4.4 Sistemas de coordenadas

- En la estructura de árbol, hacer clic con el botón derecho en **Datos de bases de coordenadas/herramientas** y haga clic en **Añadir base de coordenadas/herramienta** para mostrar la ventana de diálogo **Crear base de coordenadas/herramienta**. En **Nombre de dato de base de coordenadas/herramienta** aparece una propuesta de nombre. Completar la ventana de diálogo **Crear base de coordenadas/herramienta** con los datos de nuestra pinza de manera análoga a como se hizo al crear la herramienta. A continuación clic en **Aceptar**.



6.4.5 Dependencias

- En la estructura de árbol, hacer clic con el botón derecho en **Dependencia** y a continuación clic en **Añadir dependencia** para abrir la ventana de diálogo **Crear dependencia**. Completar la ventana de diálogo **Crear dependencia** haciendo que J1=J2, es decir, el moviento del eje 1 será igual al eje dos, (ambos dedos se acercarán o alejaran a la vez) y a continuación clic en **Aceptar**.



6.4.6 Compilación

Si todos los nodos son válidos, compilar el mecanismo.

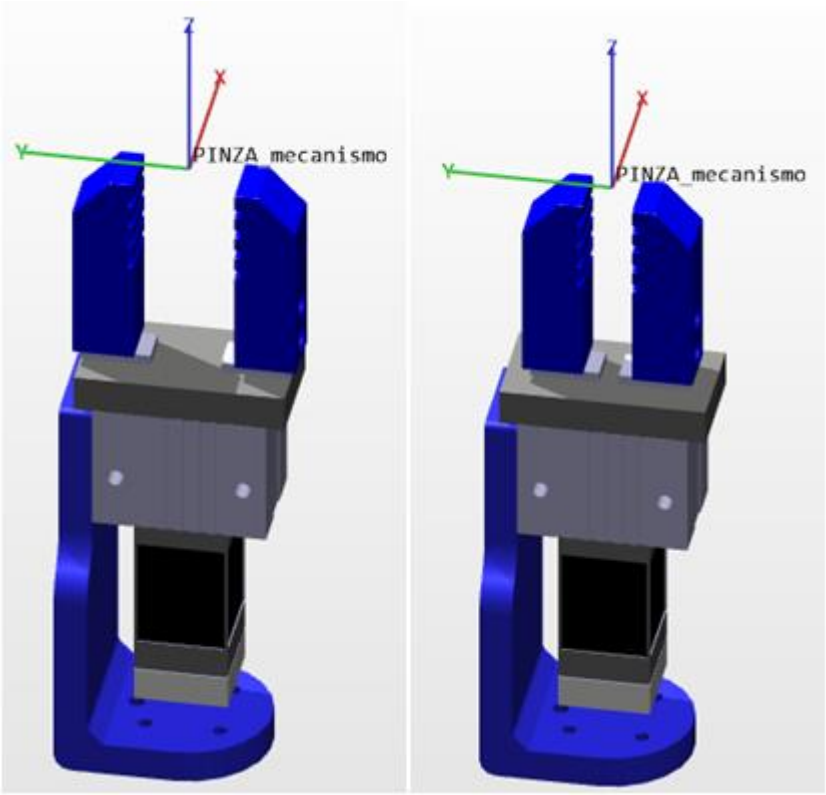
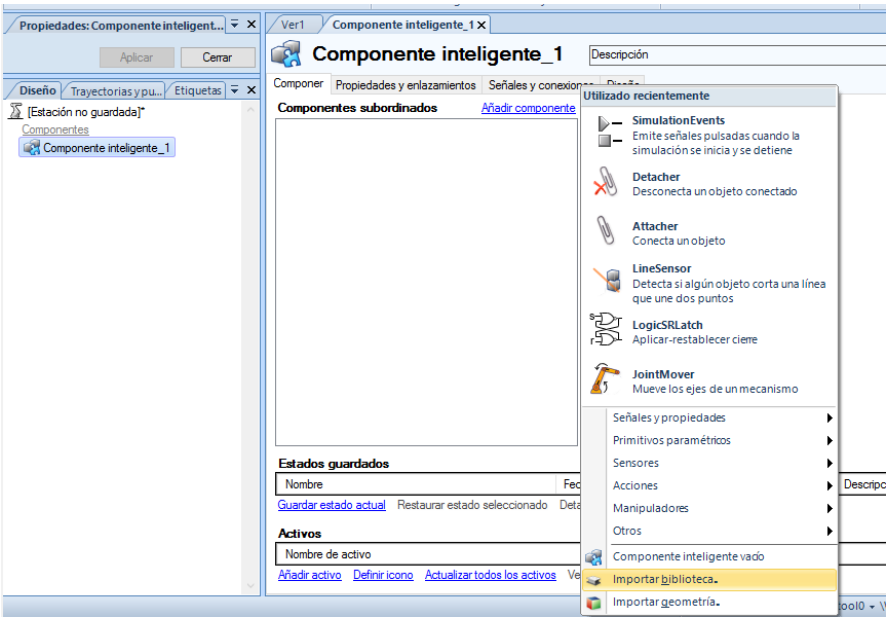


Figura 59: Mecanismo en sus dos posiciones

6.5 Creación de un Smart Component (Componente Inteligente)

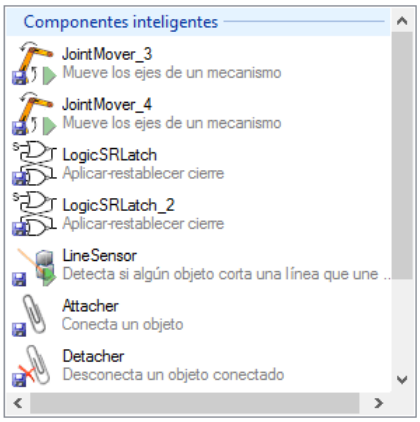
Un componente inteligente es un objeto de RobotStudio (con o sin representación gráfica en 3D) que presenta el comportamiento que puede implementarse mediante la programación de ciertas lógicas.

El primer paso es hacer click, dentro de la pestaña Modelado, en **Componente Inteligente**. Cambiaremos el nombre por defecto y le pondremos SC_PINZA_mecanismo.
En la pestaña *Añadir Componente* hacemos clik en *Importar Geometría* y seleccionamos la herramienta previamente creada, PINZA_mecanismo.



Ahora pasamos a añadir sensores y acciones que nuestro componente inteligente va a necesitar. Haciendo click de nuevo en *Añadir componente* buscamos los siguientes elementos:

- Joint Mover (2)
- Attacher (1)
- Detacher (1)
- Line Sensor (1)
- Logic SR Latch (2)
- Simulation Events (2)



El siguiente paso es acudir a la pestaña *Señales y conexiones* y hacer click en *Añadir Señales de E/S*. Las señales a añadir serán:

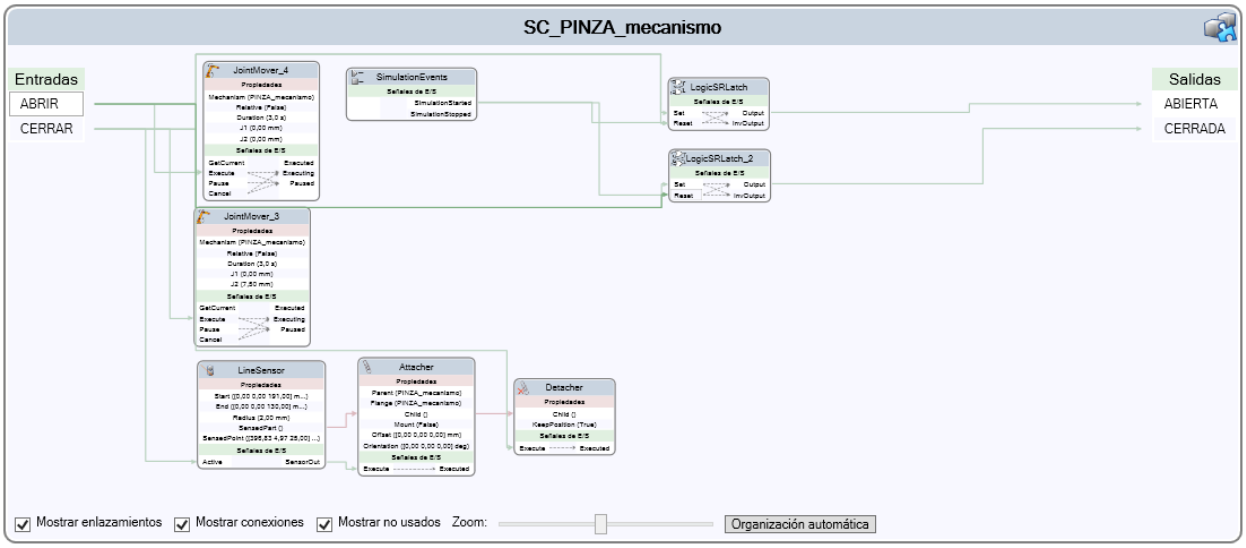
Señales de E/S

Nombre	Tipo de señal
ABRIR	DigitalInput
CERRAR	DigitalInput
CERRADA	DigitalOutput
ABIERTA	DigitalOutput

El último paso consiste en unir los bloques y configurarlos. Las conexiones serán las siguientes:

Conexiones de E/S

Objeto de origen	Señal de origen	Objeto de destino	Señal de destino
SC_PINZA_mecanismo	ABRIR	JointMover_4	Execute
SC_PINZA_mecanismo	CERRAR	JointMover_3	Execute
SC_PINZA_mecanismo	CERRAR	LineSensor	Active
LineSensor	SensorOut	Attacher	Execute
SimulationEvents	SimulationStarted	LogicSRLatch_2	Reset
SimulationEvents	SimulationStarted	LogicSRLatch	Reset
SC_PINZA_mecanismo	ABRIR	LogicSRLatch	Set
LogicSRLatch	Output	SC_PINZA_mecanismo	ABIERTA
SC_PINZA_mecanismo	CERRAR	LogicSRLatch	Reset
SC_PINZA_mecanismo	CERRAR	LogicSRLatch_2	Set
LogicSRLatch_2	Output	SC_PINZA_mecanismo	CERRADA



Los bloques y conexiones conllevarán el siguiente funcionamiento de Componente Inteligente:

- El bloque Simulation Events, hace que con cada nuevo inicio de simulación, los bloques LogicRSLatch, pongan la pinza en modo abierta.
- Al activarse la señal de entrada ABRIR:
    - Se activan el bloque JointMover, configurados para mover el eje del mecanismo de la pinza, el cual llevará dicho eje a la posición abierta.
    - Se activa el bloque Detacher, el cual hará que se suelte cualquier pieza que se tuviese agarrada en ese momento.
    - Se hace un set a uno de los bloques LogicRSLatch, el cual está conectado a la señal de salida ABIERTA, informando al sistema de que la pinza se encuentra abierta.
    - Se hace un reset al otro bloque LogicRSLatch, el cual está conectado a la señal de salida CERRADA, reseteando esta señal.

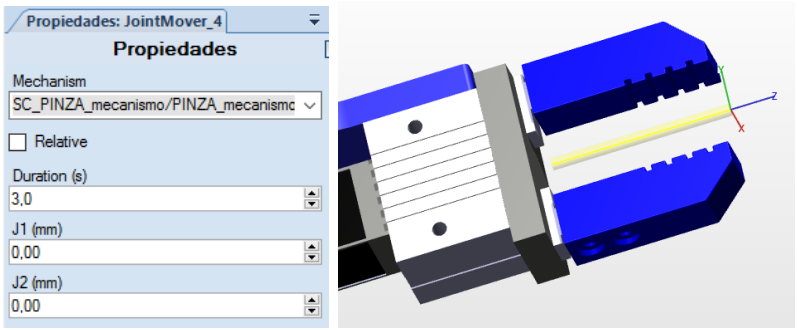


Figura 60: Propiedades del JointMover para ABRIR

- Al activarse la señal de entrada CERRAR:
    - Se activan el bloque JointMover, configurados para mover el eje del mecanismo de la pinza, el cual llevará dicho eje a la posición cerrada en 3 segundos.
    - Se activa el bloque LineSensor. Dicho bloque es un sensor de línea que debe colocarse entre los dedos de la pinza. Toda geometría que cruce dicho sensor, será enviada por el puerto de salida hacia el objeto de destino Attacher.
    - Se activa el bloque Attacher, el cual hará que se agarre la pieza detectada por el sensor de línea.
    - Se hace un set a uno de los bloques LogicRSLatch, el cual está conectado a la señal de salida CERRADA, informando al sistema de que la pinza se encuentra cerrada.
    - Se hace un reset al otro bloque LogicRSLatch, el cual está conectado a la señal de salida ABIERTA, reseteando esta señal.

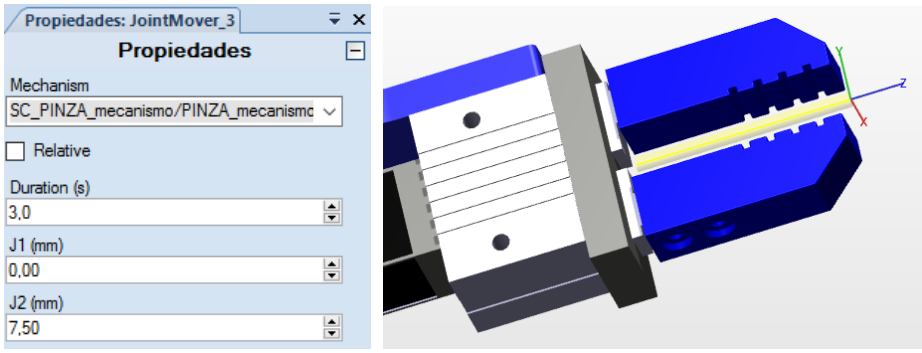
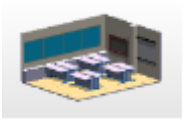
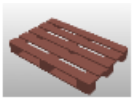
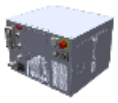


Figura 61: Propiedades del JointMover para CERRAR

6.6 Creación de la estación de trabajo

6.6.1 Colocación de las geometrías

Al crear una nueva estación, lo primero que debemos hacer es acudir a la pestaña *Importar Biblioteca, Biblioteca de Usuario* o en el caso del IRB120, *Biblioteca ABB,* y añadir las siguientes geometrías en las coordenadas específicas.

Objeto	Coordenadas	Imagen
LABORATORIO	Referencia: Mundo Posición X,Y,Z (mm): 0,00 0,00 0,00 Orientación (deg): 0 0,00 0,00	 LABORATORIO
Euro Pallet	Referencia: Mundo Posición X,Y,Z (mm): 6416,22~ 1201,42 1096,18~ Orientación (deg): 0,00 0 90,00	 Euro Pallet
IRB120	Referencia: Mundo Posición X,Y,Z (mm): 5925,00 1800,00 1240,17~ Orientación (deg): 0,00 0 0,00	 IRB 120
IRC5 Compact	Referencia: Mundo Posición X,Y,Z (mm): 6040,00 600,00 1096,18~ Orientación (deg): 0,00 0,00 90,00	 IRC5 Compact
Pinza	Anclar a IRB120 tras añadir el controlador	 SC_PINZA_mecanismo
Marcadores	Referencia: Mundo Posición X,Y,Z (mm): 6341,24 1507,48 1241,00 Referencia: Mundo Posición X,Y,Z (mm): 6341,24 1707,48 1241,00 Referencia: Mundo Posición X,Y,Z (mm): 6141,24 1507,48 1241,00	 Marcador

6.6.2 Añadir controlador

A continuación, nos dirigimos a la pestaña *Sistema Robot* y creamos *Desde diseño* un nuevo sistema. No hay que olvidar que hay que clicar en *Opciones* y añadir el bus *DeviceNet* así como cambiar el idioma.

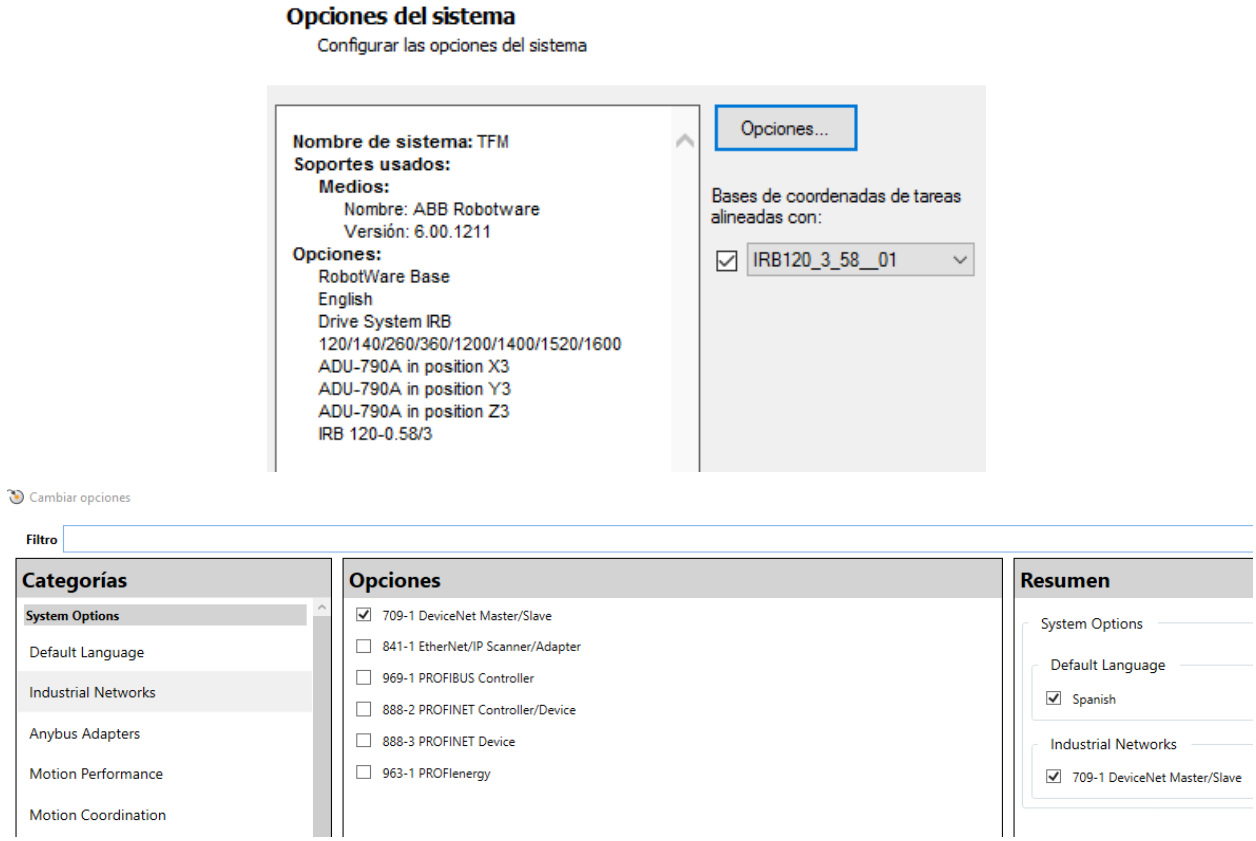


Figura 62: Opciones del controlador virtual

6.6.3 Añadir herramienta

El sistema ya se encuentra en disposición de acoplar una herramienta a la muñeca del robot. Para ello, desde la *Biblioteca de usuario* añadimos el **SC_PINZA_mecanismo** y arrastramos **el mecanismo que contiene dicho SC, es decir, PINZA_mecanismo** hasta el IRB120 para actualizar posteriormente su posición. El sensor de línea debe estar "atado a la herramienta" pero sin actualizar su posición. Es importante que seamos capaces de poder reorientar el robot alrededor del TCP de la herramienta.

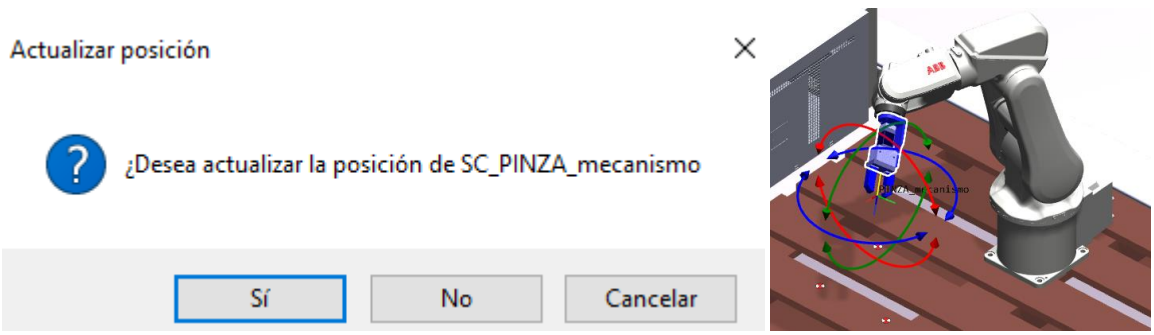
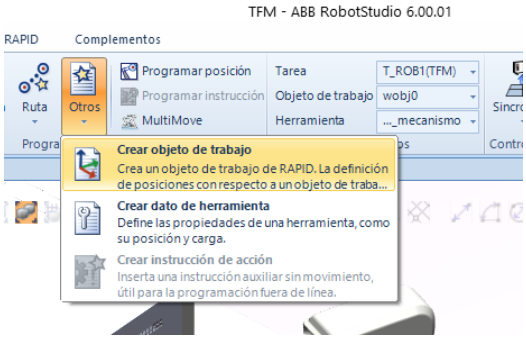


Figura 63: Actualización de la herramienta a la muñeca del robot

6.6.4 Definición del objeto de trabajo

Para trabajar de una forma más cómoda, vamos a definir un Objeto de Trabajo. Dicho sistema de referencia estará ubicado en el Pallet de madera, por lo que vamos a llamarle *Pale*. Para crear un objeto de trabajo hacemos click como muestra la imagen:



Desde la pestaña de *Trayectorias y Puntos* hacemos click derecho sobre *Pale* para fijar su posición.

Como podemos apreciar, el objeto de trabajo creado se encuentra sobre el Pallet real, los ejes X e Y coinciden en dirección hacia los marcadores antes colocados para ayudarnos a referenciarnos, y el eje Z está orientado hacia arriba.

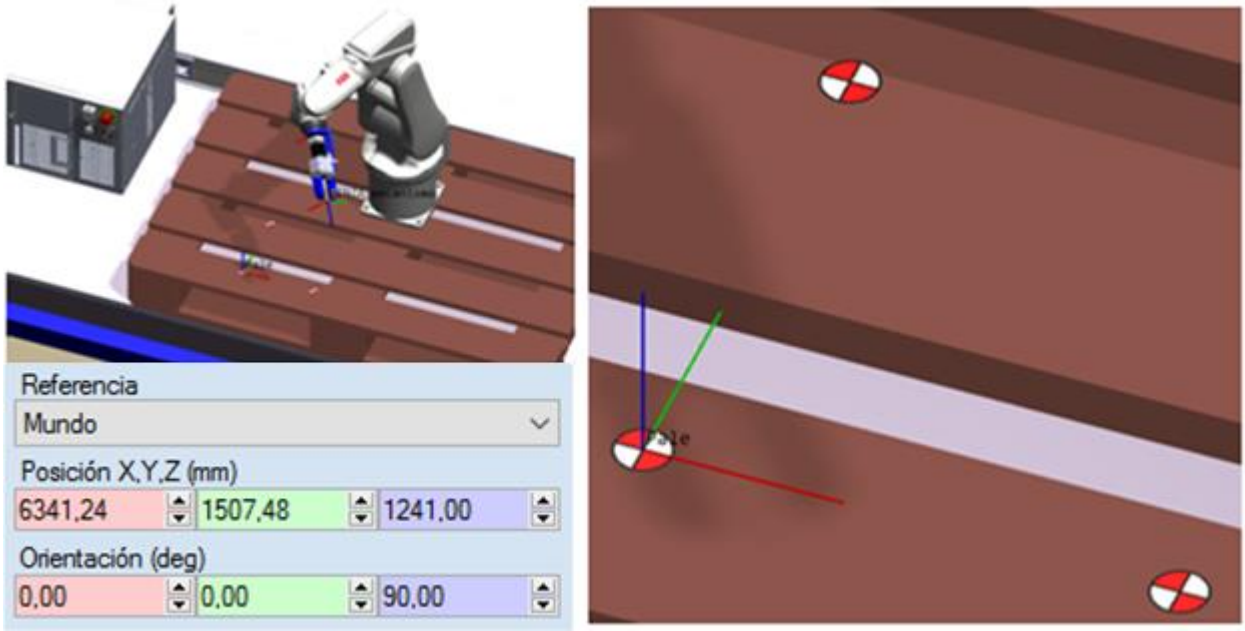
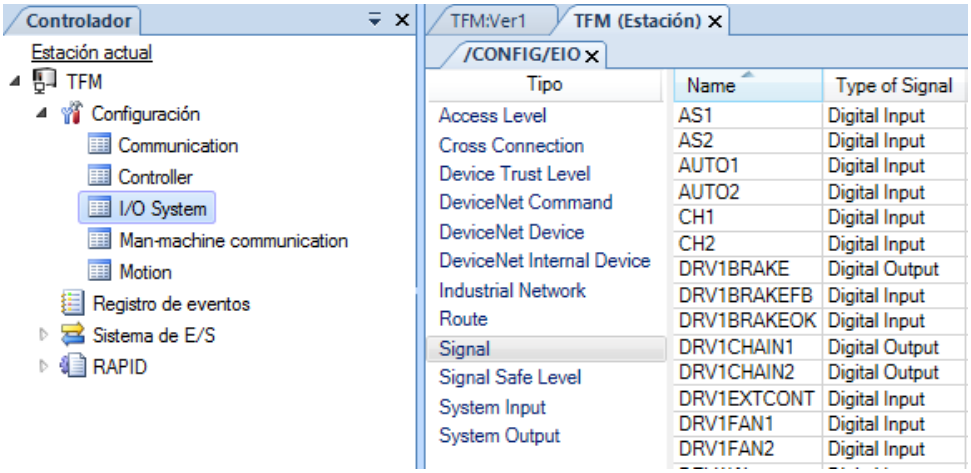


Figura 64: Objeto de trabajo *Pale*

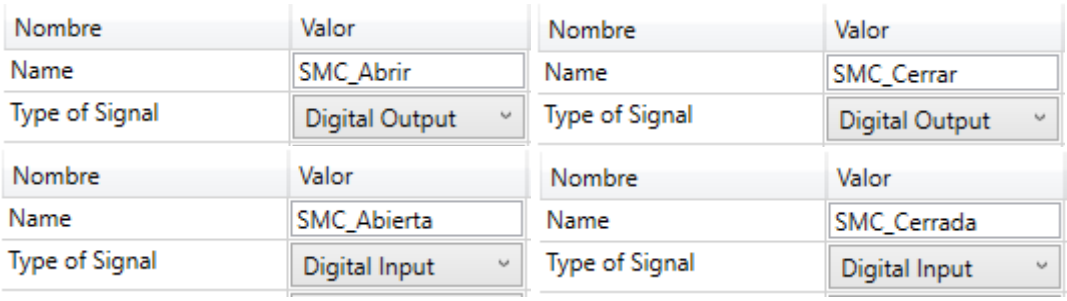
6.6.5 Creación de las señales entrada/salida y lógica de estación

El siguiente paso es crear las señales de entrada y salida del controlador que van a controlar la Pinza, en el caso virtual, controlaran el funcionamiento del Componente Inteligente.

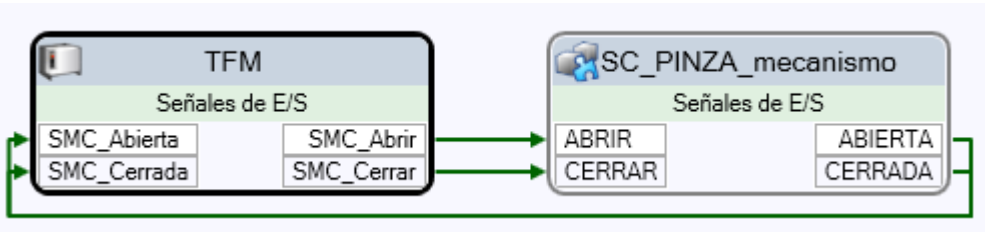
Nos dirigimos a la pestaña *RAPID* donde haremos doble click sobre *I/O System*. A continuación hacemos click en *Signal*, donde nos aparecerá la lista de todas las señales del sistema.



Al hacer click derecho sobre *Signal* se nos da la opción de crear una nueva señal. Crearemos nuestras 4 señales de control. El campo *Signal Identification Lebel* debe llevar el mismo nombre que la variable. Tras la creación de las señales, debemos *Reiniciar* el controlador.



Ahora debemos conectar las señales de entrada y salida con nuestro sistema. Para ello acudimos de *Simulación* y hacemos click en *Lógica de Estación.* En la pestaña de *Señales y conexiones* debemos añadir las 4 señales que acabamos de crear previamente haciendo click en *Añadir conexión de E/S*:

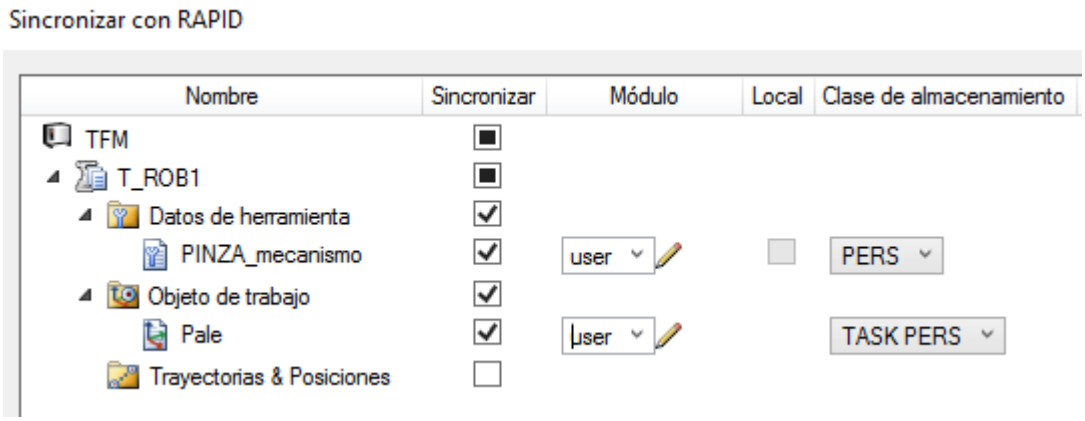


Conexiones de E/S

Objeto de origen	Señal de origen	Objeto de destino	Señal de destino
SC_PINZA_mecanismo	ABIERTA	TFM	SMC_Abierta
SC_PINZA_mecanismo	CERRADA	TFM	SMC_Cerrada
TFM	SMC_Abrir	SC_PINZA_mecanismo	ABRIR
TFM	SMC_Cerrar	SC_PINZA_mecanismo	CERRAR

6.6.6 Funciones Abrir_Pinza() y Cerrar_Pinza()

Lo primero que debemos hacer es ir a la pestaña *RAPID* y *Sincronizar* nuestra estación con todo lo previamente creado.



Como se puede apreciar, los datos de la herramienta *PINZA_mecanismo* y el objeto de trabajo *Palé* se han sincronizado con *RAPID.*

```
T_ROB1/user
1
2   MODULE user (SYSMODULE)
3
4    ! Predefined user data
5    !*********************
6
7    ! Declaration of numeric registers reg1...reg5
8   VAR num reg1 := 0;
9   VAR num reg2 := 0;
10  VAR num reg3 := 0;
11  VAR num reg4 := 0;
12  VAR num reg5 := 0;
13
14  ! Declaration of stopwatch clock1
15  VAR clock clock1;
16     PERS tooldata PINZA_mecanismo:=[TRUE,[[0,0,191],[1,0,0,0]],[1,[0,0,8],[1,0,0,0],0,1,0]];
17     TASK PERS wobjdata Pale:=[FALSE,TRUE,"",[[416.24,-292.52,7.636],[0.707106781,0,0,0.707106781]],[[0,0,0],[1,0,0,0]]];
18
19   ! Template for declaration of workobject wobj1
20   !TASK PERS wobjdata wobj1 := [FALSE, TRUE, "", [[0, 0, 0],[1, 0, 0, 0]],[[0, 0, 0],[1, 0, 0, 0]]];
21
22   ENDMODULE
```

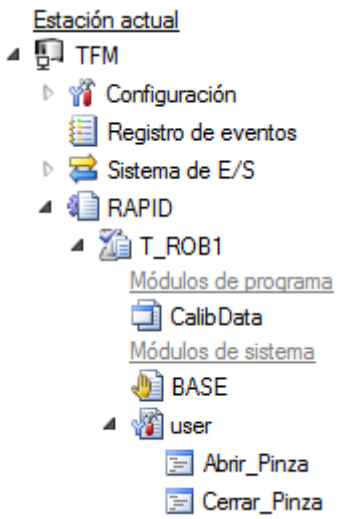
El siguiente paso es escribir unas funciones de control de alto nivel que actuarán sobre las señales que gobiernan la pinza:

```
!-------------PROGRAMA PARA ABRIR LA PINZA------------!
   PROC Abrir_Pinza()
   IF SMC_Abierta=0 THEN
       TPWrite "               Abriendo Pinza";
       SetDO SMC_Abrir,1;
       WaitTime 2;
       SetDO SMC_Abrir,0;
       WaitDI SMC_Abierta, 1;
       TPWrite "               Pinza abierta";
   ELSE
       TPWrite "               La Pinza ya está abierta";
   ENDIF
ENDPROC
!-----------------------------------------------------!
```

```
  !-------------PROGRAMA PARA CERRAR LA PINZA------------!
PROC Cerrar_Pinza()
    IF  SMC_Cerrada=0 THEN
        TPWrite "              Cerrando Pinza";
        SetDO SMC_Cerrar,1;
        WaitTime 1;
        SetDO SMC_Cerrar, 0;
        WaitDI SMC_Cerrada, 1;
        TPWrite "              Pinza cerrada";
    ELSE
        TPWrite "              La Pinza ya está cerrada";
    ENDIF
ENDPROC
  !-----------------------------------------------------!
```

Figura 65: Funciones Abrir_Pinza y Cerrar_pinza

Tras esto, debemos volver a *Sincronizar* nuestro sistema y comprobamos que nuestras dos funciones ya se encuentran en el modulo *user*.

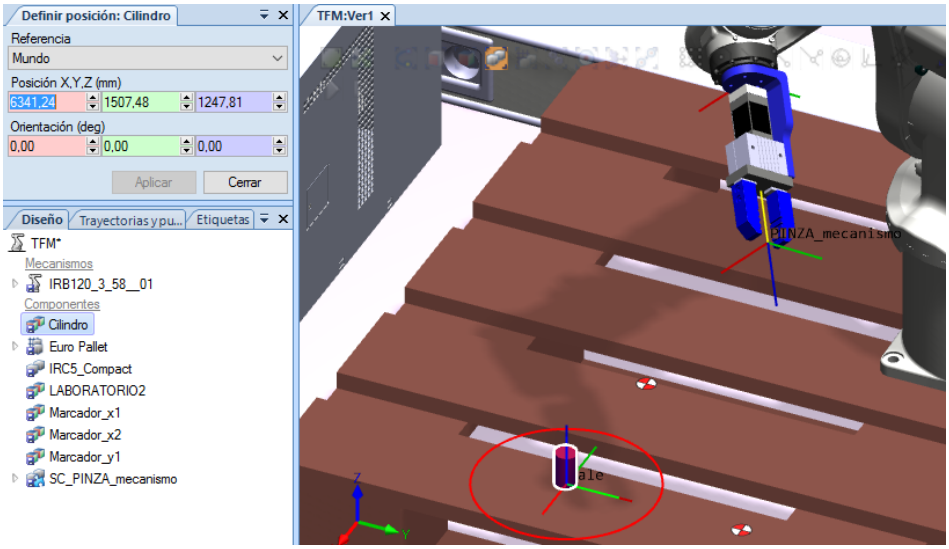


6.7 Programa de ejemplo

Una vez abierta nuestra estación, la cual cuenta ya con las geometrías posicionadas, la herramienta acoplada, y el objeto de trabajo creado, solo nos queda crear nuestra aplicación con los objetos que deseemos mover.

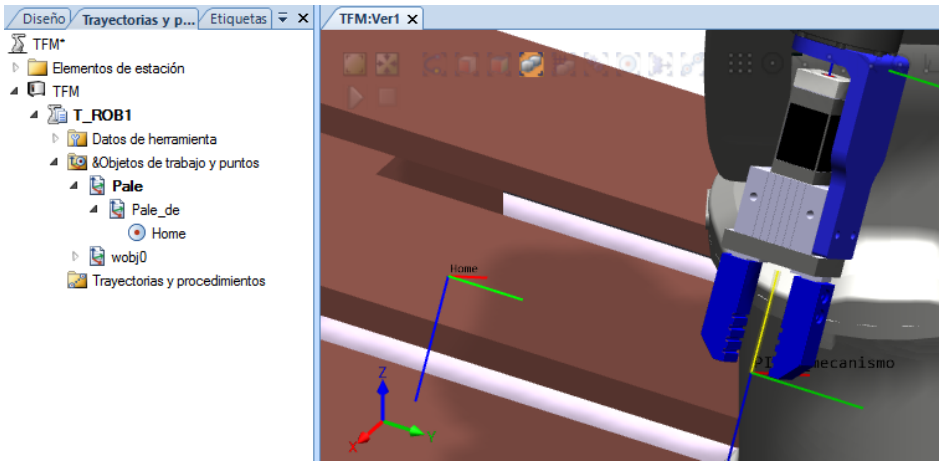
6.7.1 Introducir objetos en la estación virtual

Colocamos por ejemplo un cilindro sobre el marcador que indica la posición (0, 0, 0) del objeto de trabajo *Pale.*

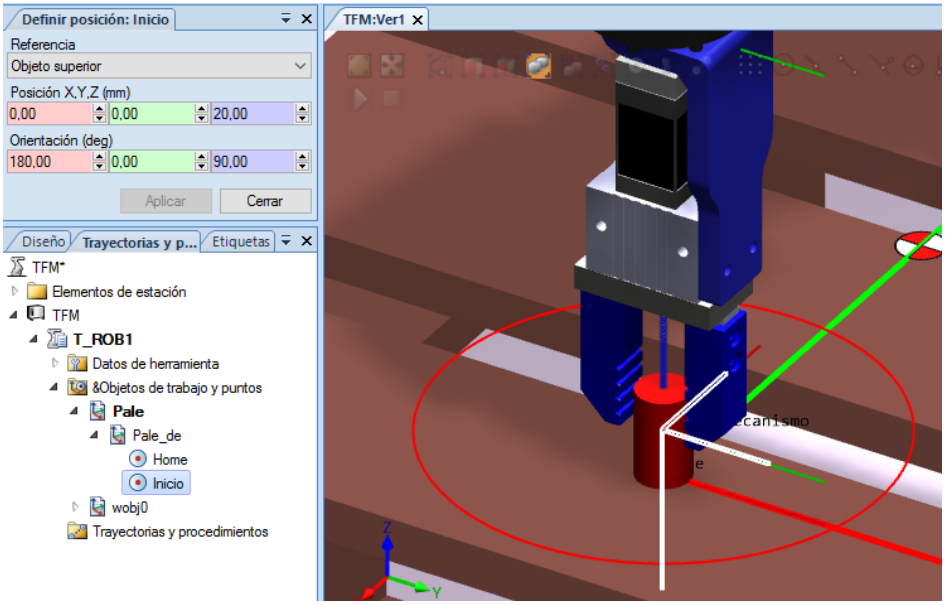


6.7.2 Programar puntos

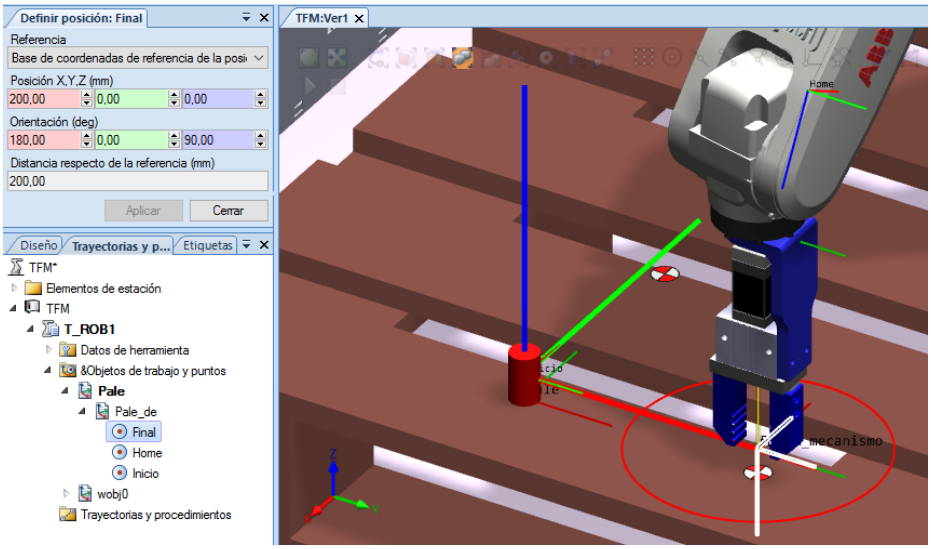
Con el robot en la posición actual, vamos a definir un punto de paso. Para ello, con las opciones Objeto de trabajo marcando la opción Pale y Herramienta marcando PINZA_mecanismo, hacemos click en *Programar posición* y la definiremos como Home.



Ahora programamos el punto *Inicio.* Referenciado al Objeto de trabajo *Pale*, 20mm por encima de este en el eje Z y rotado 180º para que sea alcanzable por el robot.



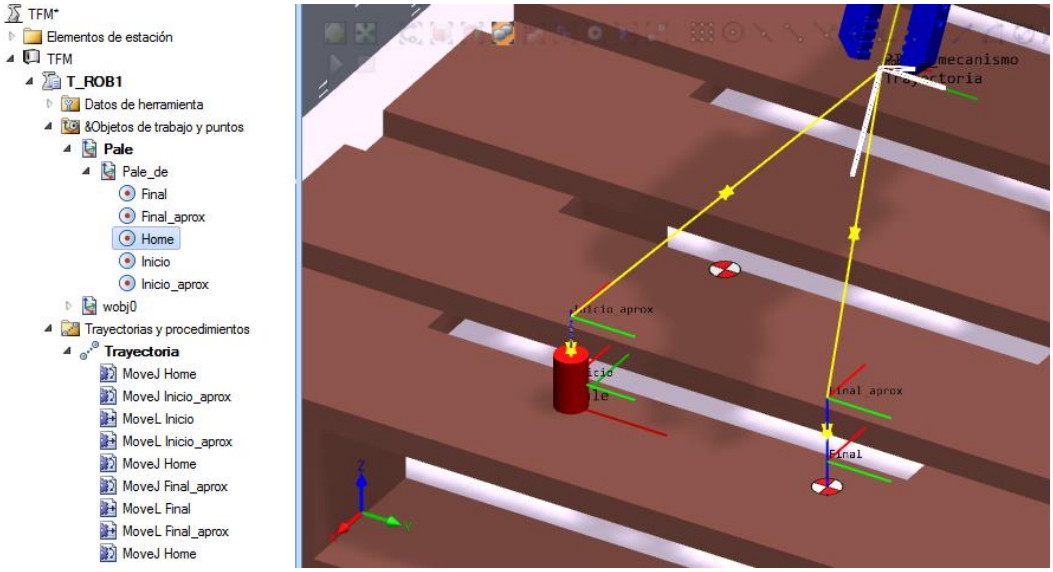
Lo mismo hacemos con el punto de destino *Final.*



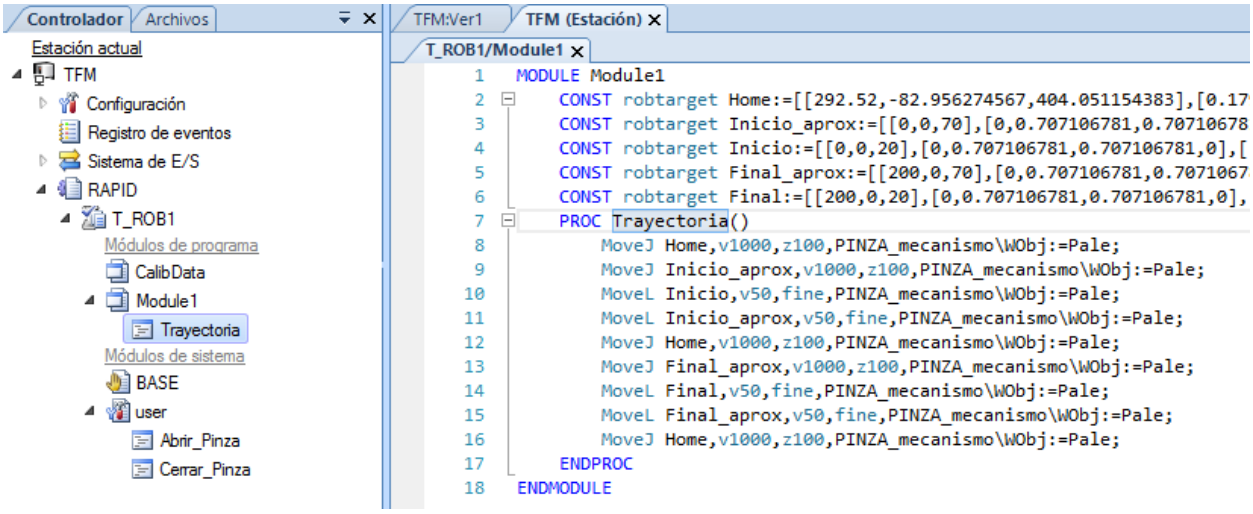
Además, hemos creado unos puntos de aproximación a los puntos anteriores para que la pinza no colisione contra nuestro objeto.

6.7.3 Programar movimientos

Para crear una trayectoria, debemos hacer clic sobre *Ruta, Trayectoria vacía.* Modificamos el nombre de la ruta y arrastramos uno a uno los puntos a los que deseamos que nuestro robot cruce.



Ahora debemos *Sincronizar* y al dirigirnos a la pestaña RAPID podemos comprobar que nuestra *Trayectoria* ha aparecido como un nuevo modulo de programa. Es importante que nos aseguremos de que todos nuestros puntos y movimientos están referenciados a la herramienta *PINZA_mecanismo* y al objeto de trabajo *Pale*.



Ahora, para comprobar de momento que funciona, sin abrir ni cerrar la pinza, podemos irnos ala pestaña *Simulacion,*hacemos click en *Configuracion de simulación,* click en T_ROB1 y donde pone *Punto de entrada* seleccionamos nuestro programa llamado *Trayectoria.* Le damos a cerrar y ya podemos darle al play en *Reproducir* para ver como se mueve nuestro robot.

El último paso es hacer uso de las funciones previamente cargadas en *user* de *Abrir_Pinza* y *Cerrar_Pinza.* Para ello, nos vamos de nuevo al código RAPID, y en los momentos en los que queramos abrir y cerrar la pinza, solo tenemos que escribir dichas funciones.

```
MODULE Module1
    CONST robtarget Home:=[[292.52,-82.956274567,404.051154383],[
    CONST robtarget Inicio_aprox:=[[0,0,70],[0,0.707106781,0.7071
    CONST robtarget Inicio:=[[0,0,20],[0,0.707106781,0.707106781,
    CONST robtarget Final_aprox:=[[200,0,70],[0,0.707106781,0.707
    CONST robtarget Final:=[[200,0,20],[0,0.707106781,0.707106781
    PROC Trayectoria()
        MoveL Home,v100,fine,PINZA_mecanismo\WObj:=Pale;
        MoveL Inicio_aprox,v100,fine,PINZA_mecanismo\WObj:=Pale;
        Abrir_Pinza;
        MoveL Inicio,v50,fine,PINZA_mecanismo\WObj:=Pale;
        Cerrar_Pinza;
        MoveL Inicio_aprox,v50,fine,PINZA_mecanismo\WObj:=Pale;
        MoveL Home,v100,fine,PINZA_mecanismo\WObj:=Pale;
        MoveL Final_aprox,v100,fine,PINZA_mecanismo\WObj:=Pale;
        MoveL Final,v50,fine,PINZA_mecanismo\WObj:=Pale;
        Abrir_Pinza;
        MoveL Final_aprox,v50,fine,PINZA_mecanismo\WObj:=Pale;
        MoveL Home,v100,fine,PINZA_mecanismo\WObj:=Pale;
        Cerrar_Pinza;
    ENDPROC
ENDMODULE
```

7.1 Continuación del programa de ejemplo

Para completar nuestro programa de ejemplo, el último paso a dar consiste en la transferencia y ejecución en el robot real. Suponiendo que hemos realizado nuestro programa en RobotStudio desde nuestro ordenador personal, debemos guardar la estación para poder llevarla al ordenador conectado al robot. Para ello, lo primero que debemos hacer es un *Pack&Go.*

Para hacer un *Pack&Go*, primero guardamos nuestra estación, para a continuación, dirigirnos a la pestaña *Archivo, Compartir, Pack&Go.*

Figura 66: Guardar como Pack&Go

Al hacer esto, hemos creado un archivo comprimido que contiene todo lo necesario para poder ejecutar nuestra aplicación en otro sistema. Al abrir dicho archivo en el ordenador conectado al robot, debemos crear un controlador real, aparte del virtual, que será donde ejecutemos nuestro programa.

Para ello, pulsamos sobre la pestaña *Controlador, Añadir controlador, Añadir controlador:*

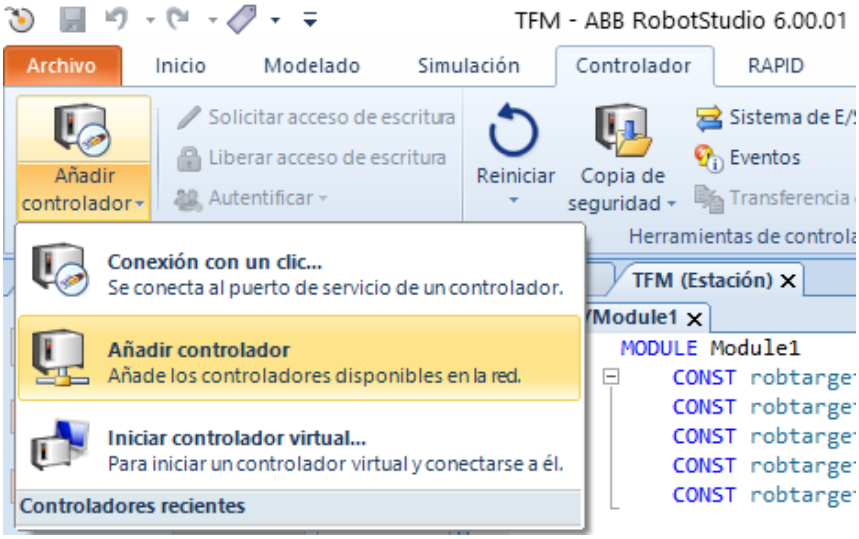
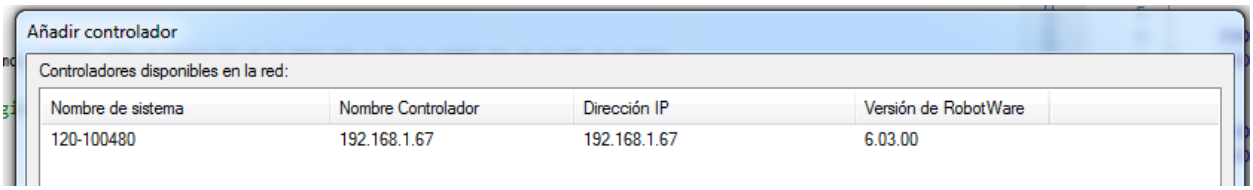


Figura 67: Añadir controlador real

Al hacer click, nos debe de aparecer el controlador real, IRC-5:



Seleccionamos dicho controlador. Ya tenemos acceso a ambos controladores, el **real** y el **virtual**:

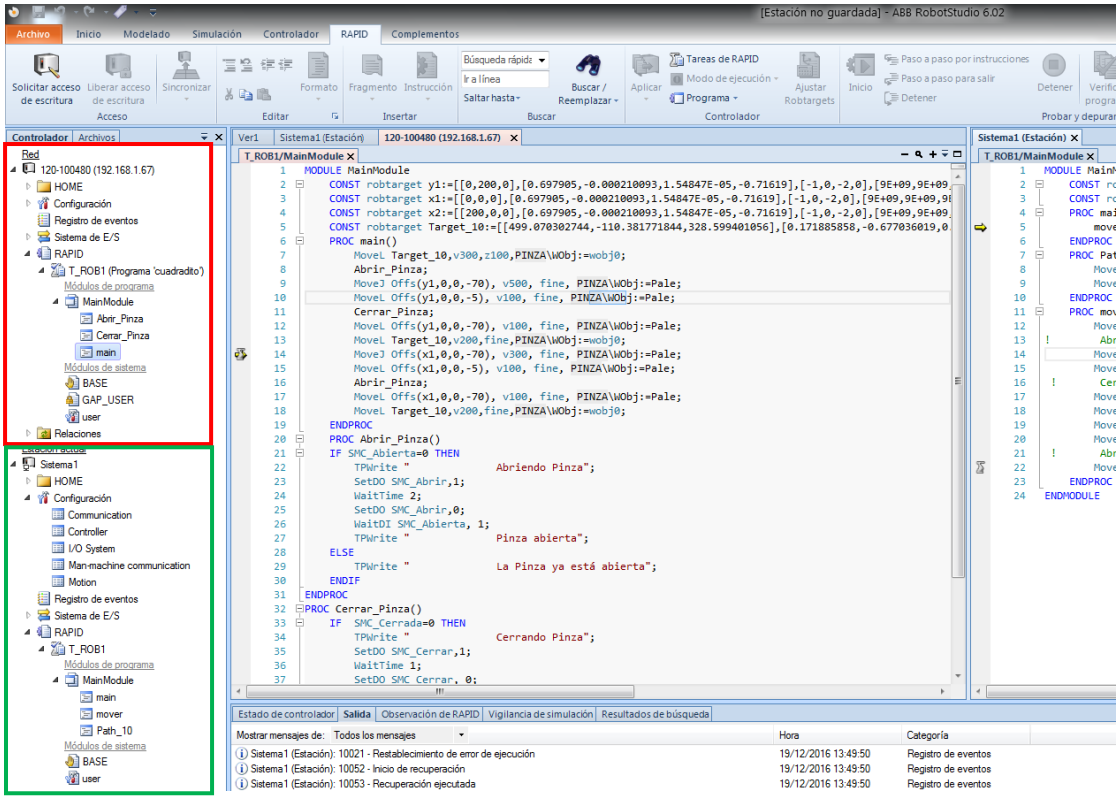
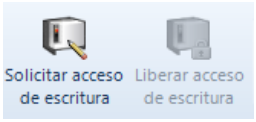
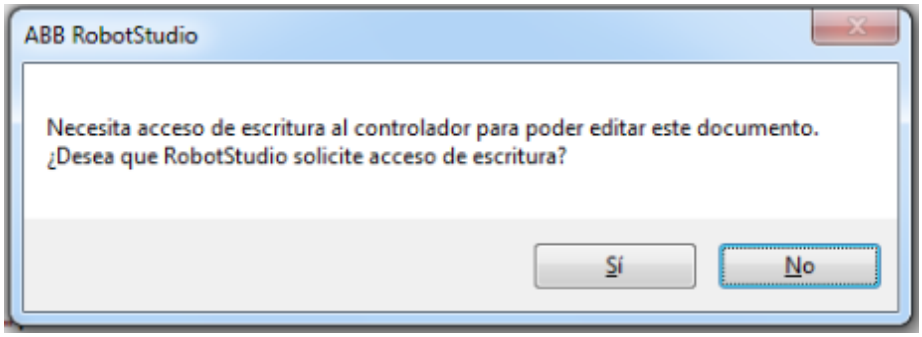
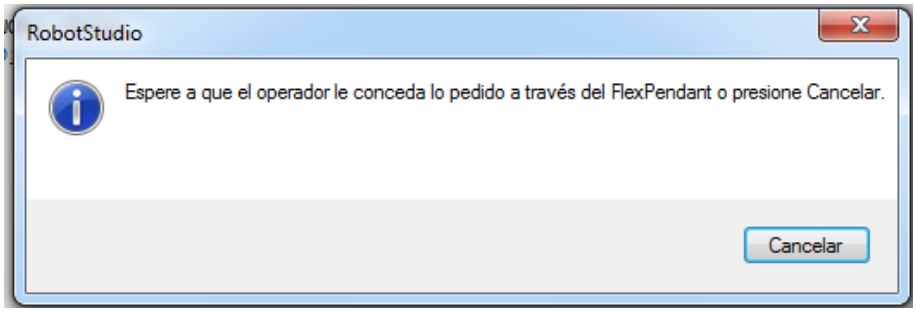


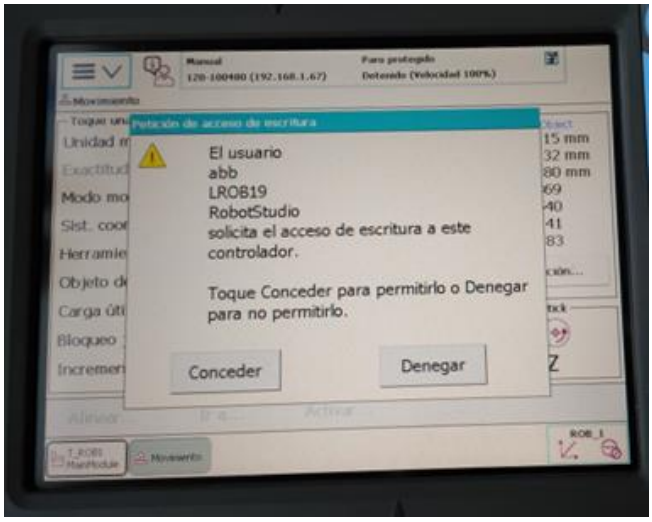
Figura 68: Controladores real y virtual

Ahora hacemos click sobre cualquier línea de código perteneciente al controlador real o directamente solicitamos permiso de escritura. Nos aparecerá un aviso que informa sobre la solicitud de permisos de escritura, que deben ser concedidos desde la Flexpendant del robot real.

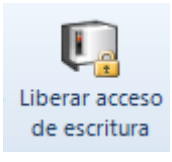








Si quisiéramos cancelar los permisos de escritura, podríamos revocarlos desde la Flexpendant o cancelarlos desde el ordenador.



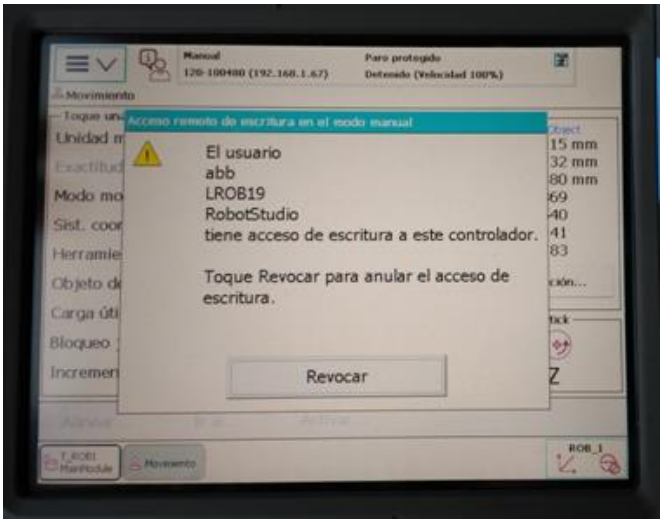


Figura 69: Proceso de enlace con la controladora real

Ahora ya podemos copiar nuestro código de un controlador a otro haciendo un simple *Copiar-Pegar*. Solo nos queda pulsar sobre el botón *Aplicar*, y nuestro programa habrá sido transferido a la controlador real del robot.

Como podemos comprobar desde la Flexpendant, el programa creado ya se encuentra listo para ser ejecutado pulsando *Play* sobre la botonera:

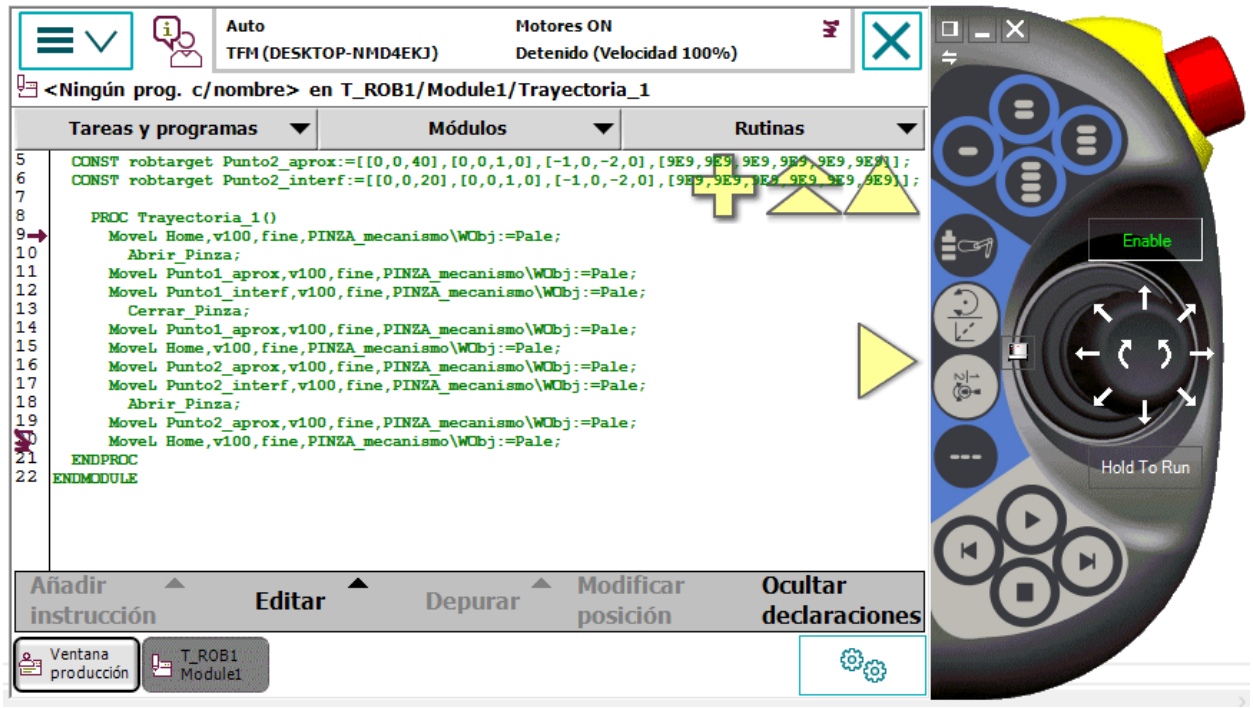


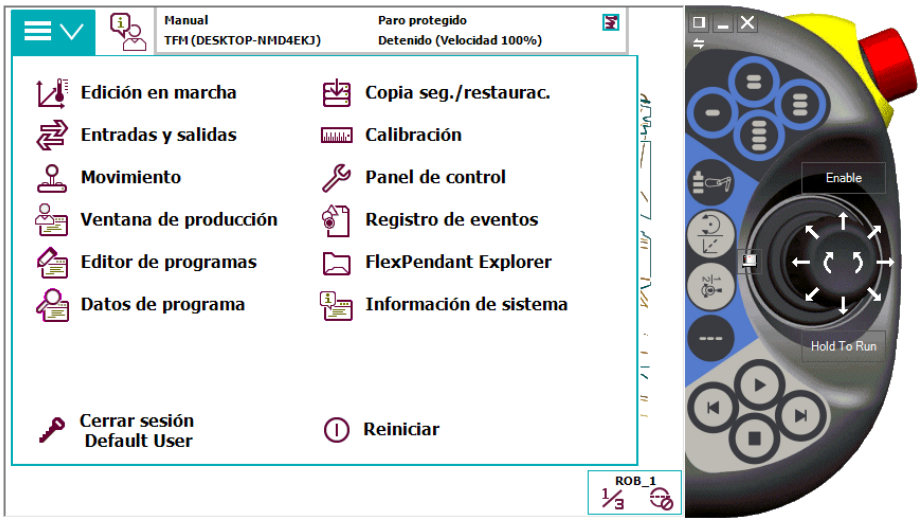
Figura 70: Programa cargado en el robot real

7.2 Definiciones y funciones a cargar en el controlador IRC5 real

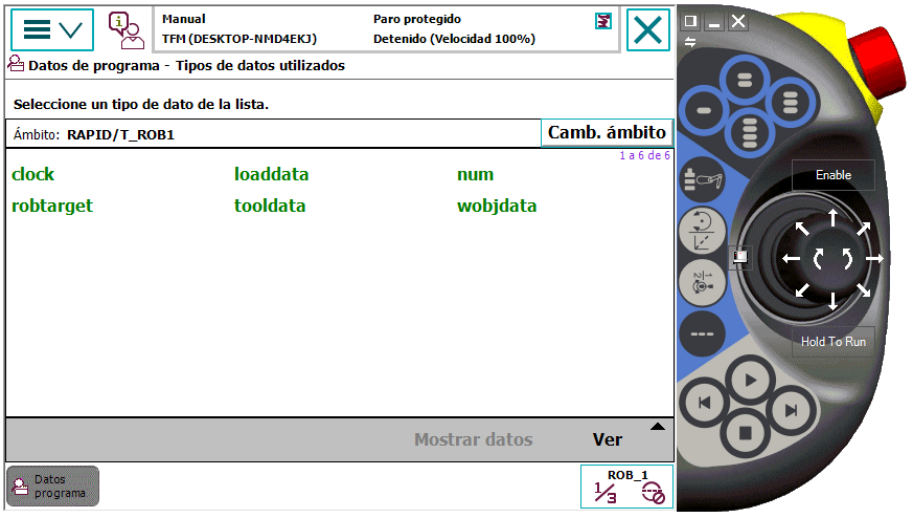
Para poder transferir programas de la controladora virtual a la controladora y el robot real, los parámetros de herramienta, objeto de trabajo, señales de entrada salida, y funciones cargadas en el módulo de usuario deben ser configurados.

7.2.1 Definición de la herramienta

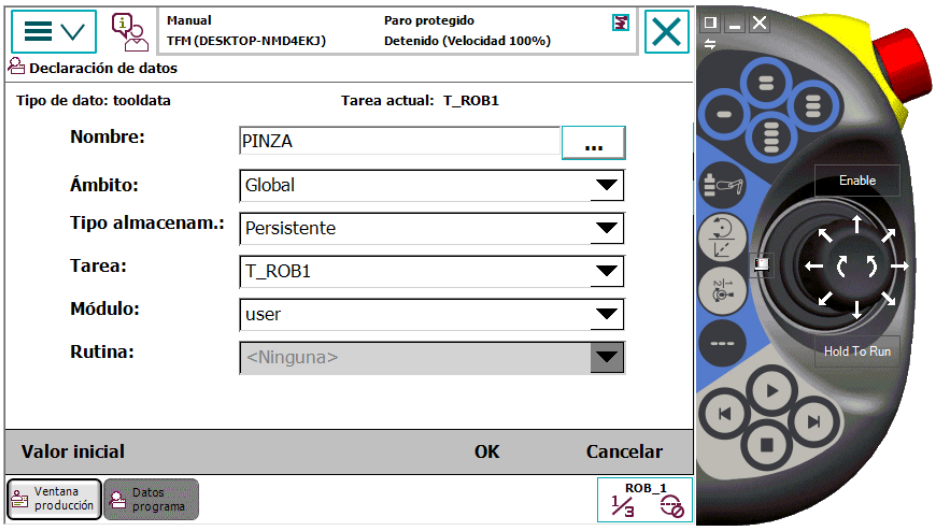
Para definir una herramienta desde el Flexpendant, debemos pulsar sobre *Datos de Programa:*



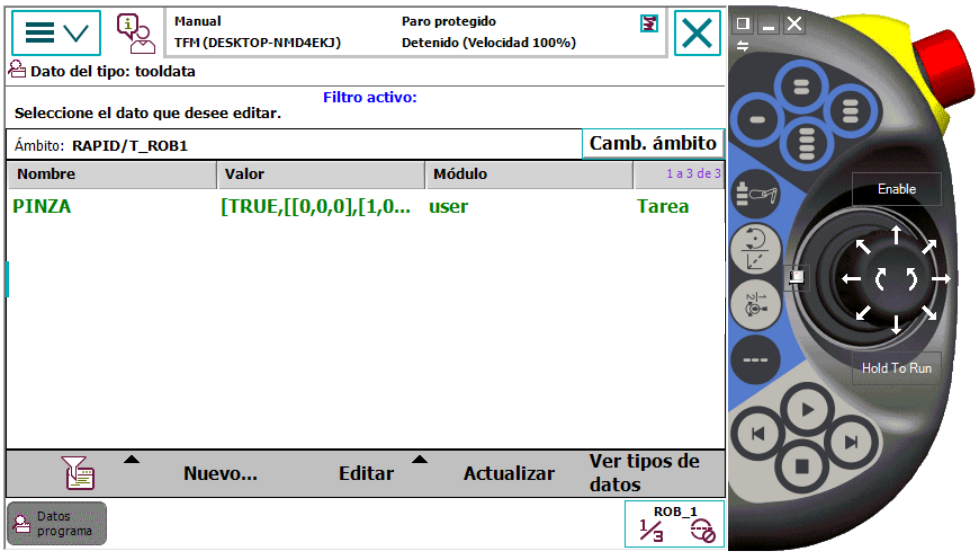
tooldata:



Y hacer clik en *Nuevo*. Definiremos nuestra herramienta con un *Nombre*, de tipo *Persistente*, y la almacenaremos en módulo *user*:



Pulsamos *OK* y en la ventana *tooldata* se nos crea una nueva herramienta con el nombre especificado anteriormente.



Llegados a este punto, contamos con dos maneras de definir el TCP de la herramienta.

- Método directo.
- Método de los 4 puntos

7.2.1.1 Método directo

Para llevar a cabo la configuración directa de la herramienta, debemos conocer las dimensiones de la misma, así como la orientación del TCP con respecto a las coordenadas del eje final del robot. Introducir dicho datos, hacemos click sobre el nombre de la herramienta (PINZA_mecanismo) e introducimos los datos necesarios.

En nuestro caso concreto, los únicos datos a introducir son el **peso (500g aprox)** y una traslación en el **eje z** de **191mm.**

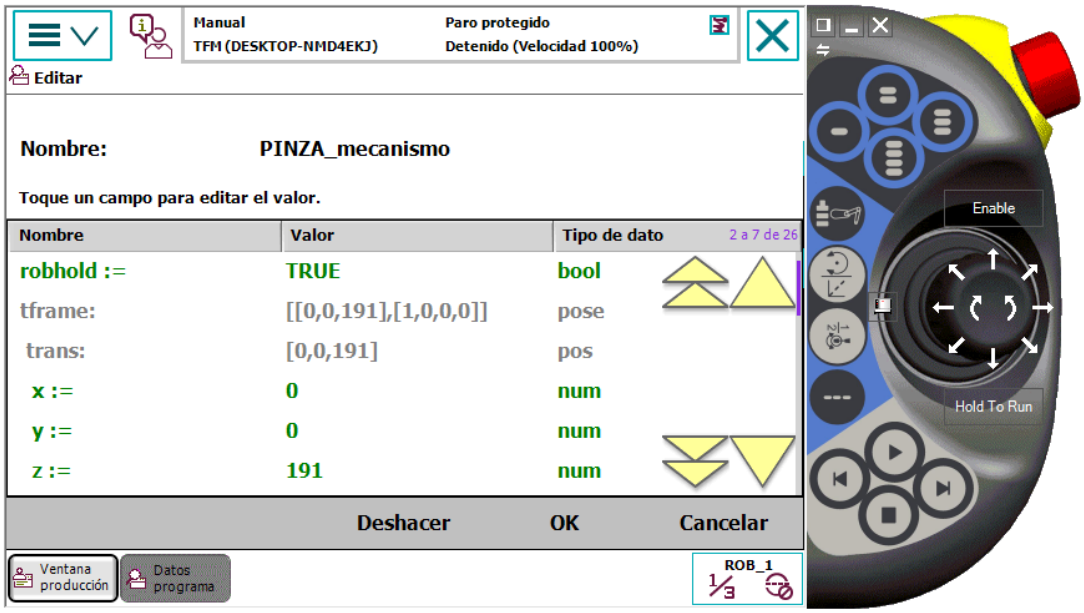


Figura 71: Configuración directa de la herramienta

7.2.1.2 Método de los 4 puntos

Se necesita en primer lugar, un punto de referencia fijo. Existen tres métodos diferentes que pueden usarse a la hora de definir la base de coordenadas de la herramienta. Los tres requieren que defina las coordenadas cartesianas del punto central de la herramienta. La diferencia está en la forma de definir la orientación.

Si desea...	...a continuación, seleccione
Utilizar la misma orientación que la de la placa de montaje del robot	TCP (orient. predet.)
cambiar la orientación en el eje Z	TCP&Z
cambiar la orientación en los ejes X y Z	TCP&Z,X

Tabla 17: Métodos de introducción de TCP

En este procedimiento se describe cómo seleccionar el método utilizado a la hora de definir la base de coordenadas de la herramienta.

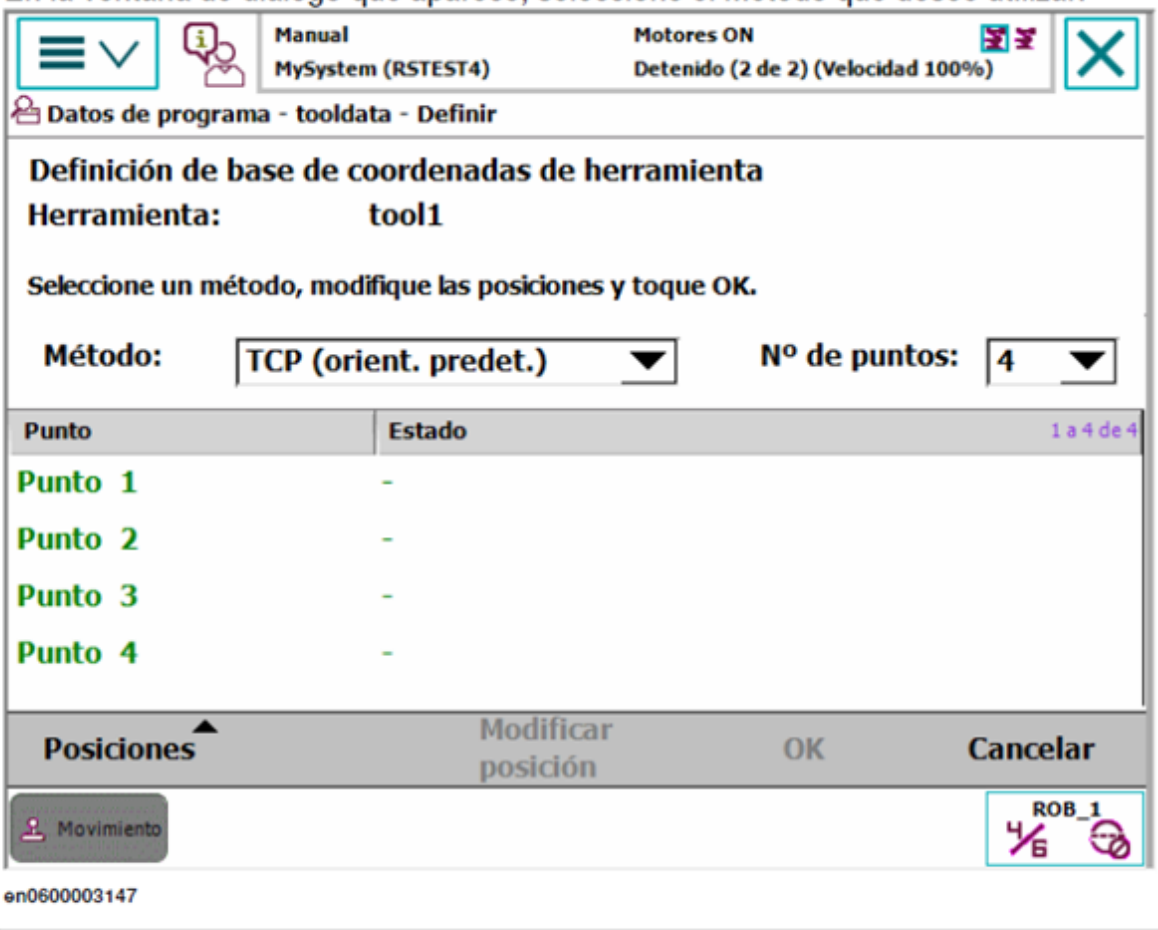
	Acción
1	En el menú ABB, toque **Movimiento**.
2	Toque **Herramienta** para ver la lista de herramientas disponibles.
3	Seleccione la herramienta que desee definir.
4	En el menú Editar, toque **Definir**.
5	En la ventana de diálogo que aparece, seleccione el método que desee utilizar. 
6	Seleccione el número de puntos de aproximación a usar. Normalmente basta con 4 puntos. Si elige más puntos para obtener un resultado más exacto, debe poner el mismo cuidado al definir cada uno de ellos.

Tabla 18: Pasos a seguir para introducir TCP

	Acción	Información
1	Mueva el robot hasta una posición adecuada, A, para el primer punto de aproximación.	Utilice incrementos pequeños para posicionar con exactitud la punta de la herramienta lo más cerca posible del punto de referencia.
2	Toque **Modificar posición** para definir el punto.	
3	Repita los pasos 1 y 2 con cada punto de aproximación que desee definir, posiciones B, C y D.	Aléjese del punto mundo fijo para conseguir los mejores resultados. Si sólo cambia la orientación de la herramienta, no obtendrá unos resultados tan adecuados.
4	Si el método que está utilizando es TCP & Z o TCP & Z, X, también es necesario definir la orientación.	Siga las instrucciones de *Cómo definir puntos de elongador en la página 192*.
5	Si por algún motivo desea repetir el procedimiento de calibración descrito en los pasos del 1 al 4, toque **Posiciones** y a continuación **Restablecer todo**.	
6	Cuando todos los puntos estén definidos, puede guardarlos en un archivo, lo que permite reutilizarlos más tarde. En el menú **Posiciones**, toque **Guardar**.	
7	Toque **OK**. En este momento aparece la ventana de diálogo **Resultado de cálculo**, que pide que cancele o confirme el resultado antes de escribirlo en el controlador.	Para obtener más información, consulte *¿Es suficientemente bueno el resultado calculado? en la página 192*

Figura 72: Posicionamiento en los 4 puntos

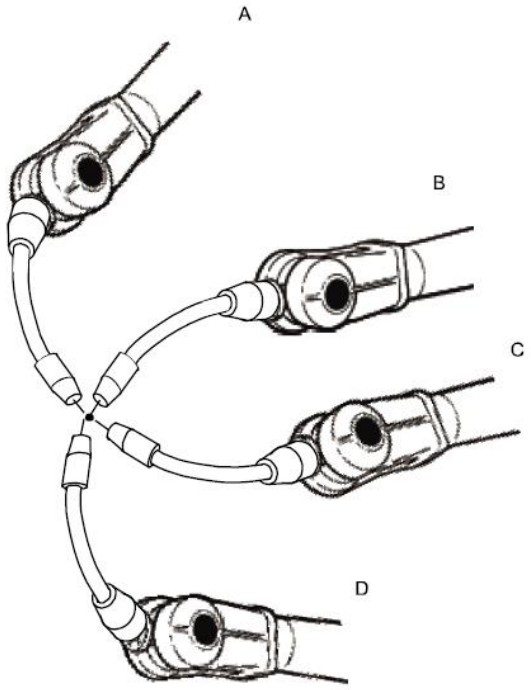


Figura 73: 4 orientaciones distintas

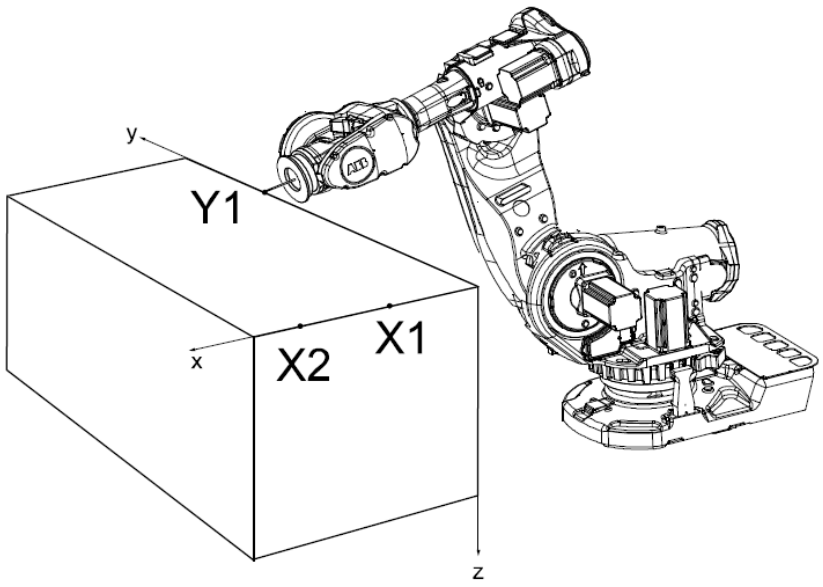
7.2.2 Definición del objeto de trabajo

Para crear un nuevo objeto de trabajo debemos seguir los siguientes pasos:

	Acción
1	En el menú **ABB**, toque **Movimiento**.
2	Toque **Objeto de trabajo** para ver la lista de objetos de trabajo disponibles.
3	Toque **Nuevo...** para crear un nuevo objeto de trabajo.
4	Toque **OK**.

Tabla 19: Introducción del objeto de trabajo

La definición de un objeto de trabajo implica que se usa el robot para apuntar a su ubicación. Esto se hace mediante la definición de tres posiciones: dos en el eje X y una en el eje Y. El eje X pasará por los puntos X1-X2 y el eje Y pasará por Y1.



	Acción	Información
1	En el menú **Método de usuario**, toque **3 puntos**.	
2	Presione el dispositivo de habilitación y mueva el robot hasta el primer punto (X1, X2 o Y1) que desee definir.	El uso de una distancia elevada entre X1 y X2 resulta preferible y permite obtener una definición más exacta.
3	Seleccione el punto en la lista.	
4	Toque **Modificar posición** para definir el punto.	
5	Repita los pasos del 2 al 4 con los demás puntos.	

Tabla 20: Introducción de los puntos X1, X2, Y1

7.2.3 Definición de las entradas y salidas

Las señales se configuran desde *Entradas y salidas.*

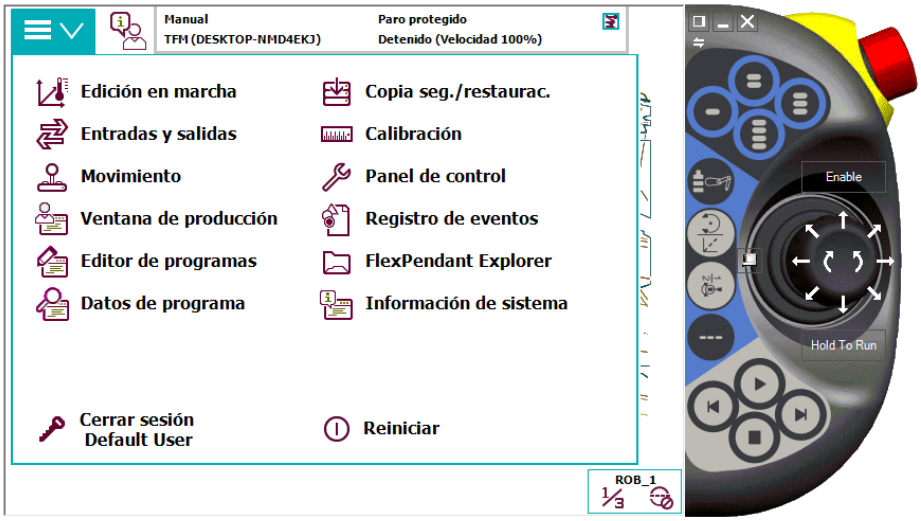


Figura 74: Entradas y salidas

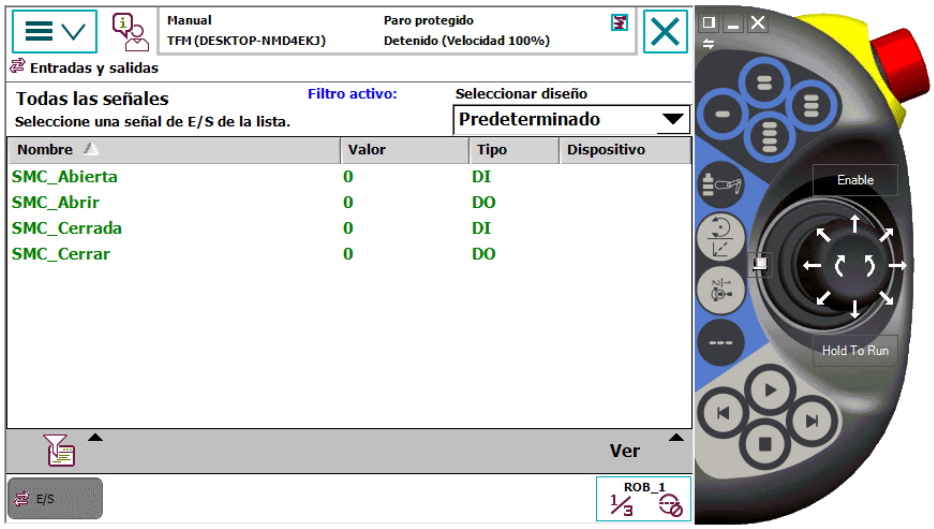
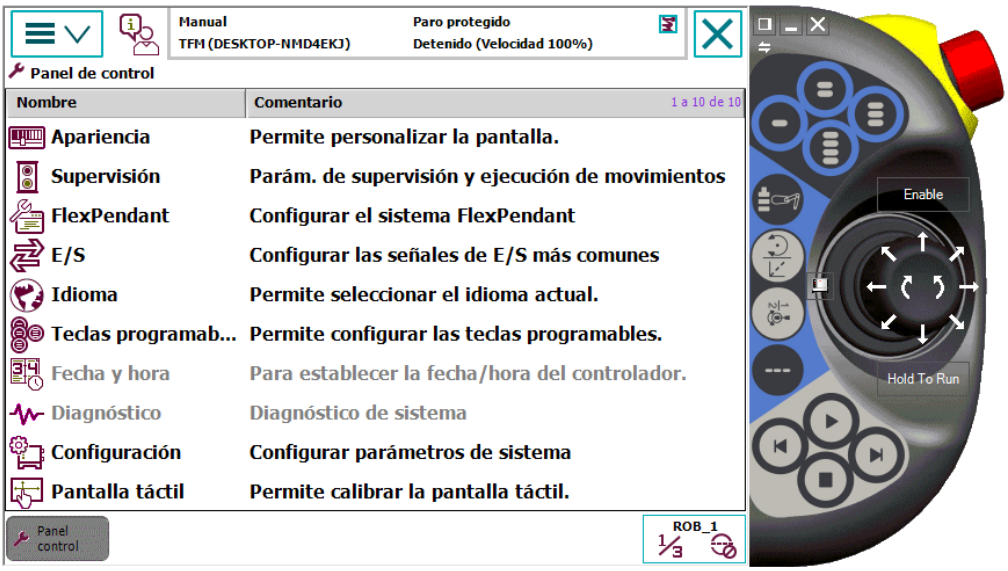


Figura 75: Entradas y salidas digitales

7.2.4 Definición de las teclas programables

Otra forma de abrir y cerrar la pinza, pasa por actuar directamente sobre las señales de entrada y salida previamente definidas. La forma de actuar directamente en este caso, consiste en pulsar las teclas programables que se encuentran en la Flexpendant.

Para ello, nos vamos a *Panel de Control, Teclas programables:*



Una vez aquí, configuramos la tecla 1/2 para que envíe active la señal de salida SMC_Abrir / SMC_Cerrar mientras se mantenga presionada dicha tecla:

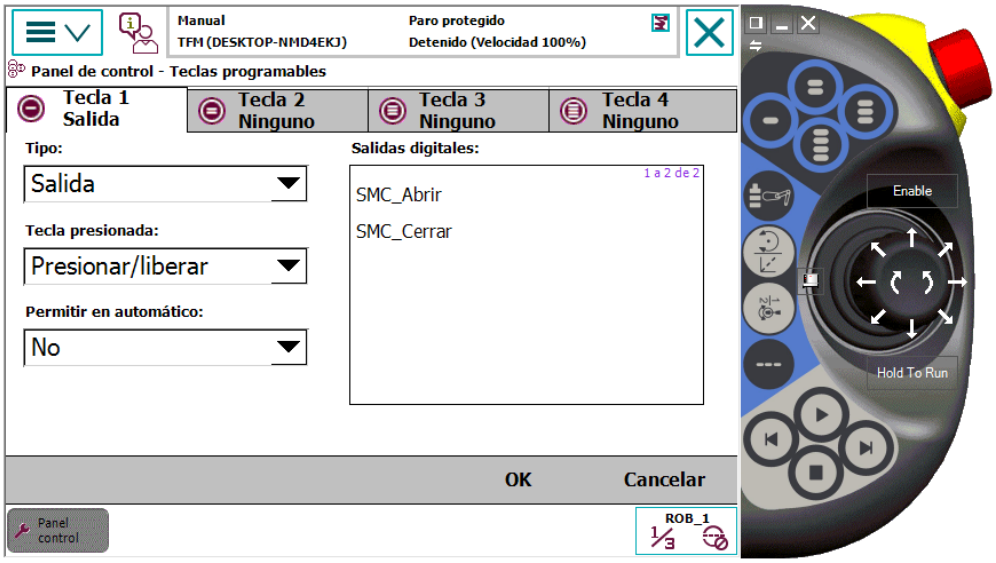
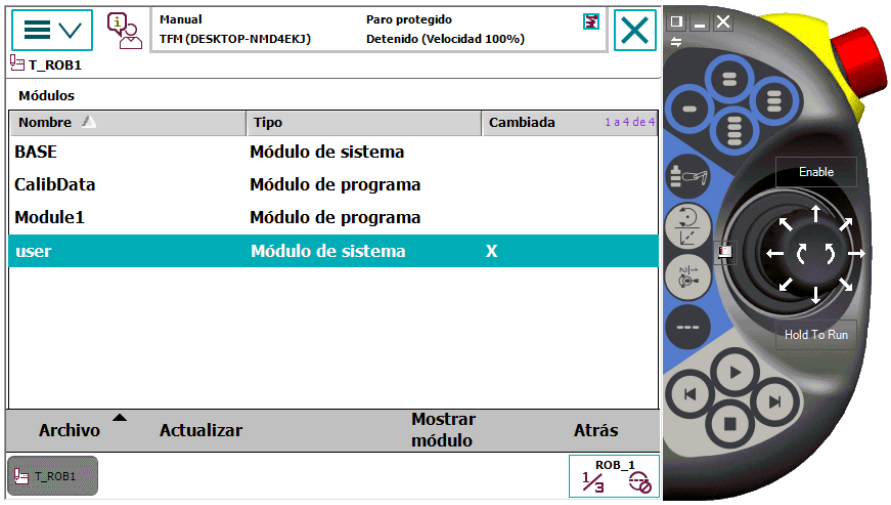
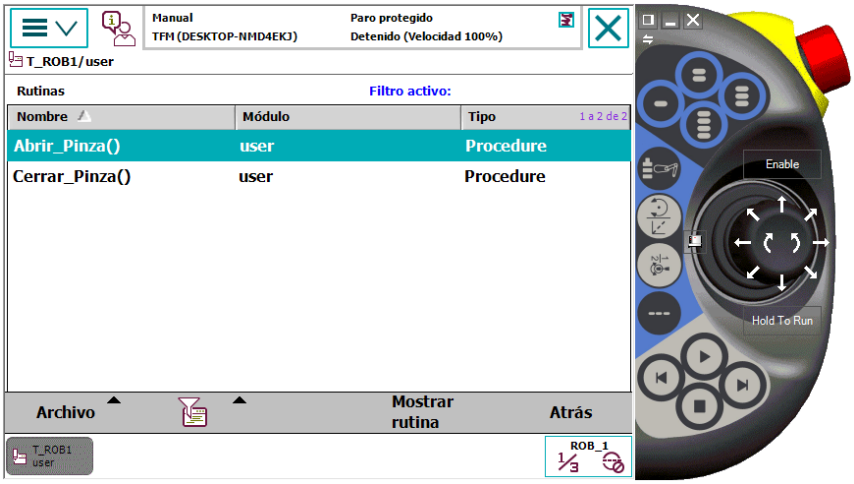


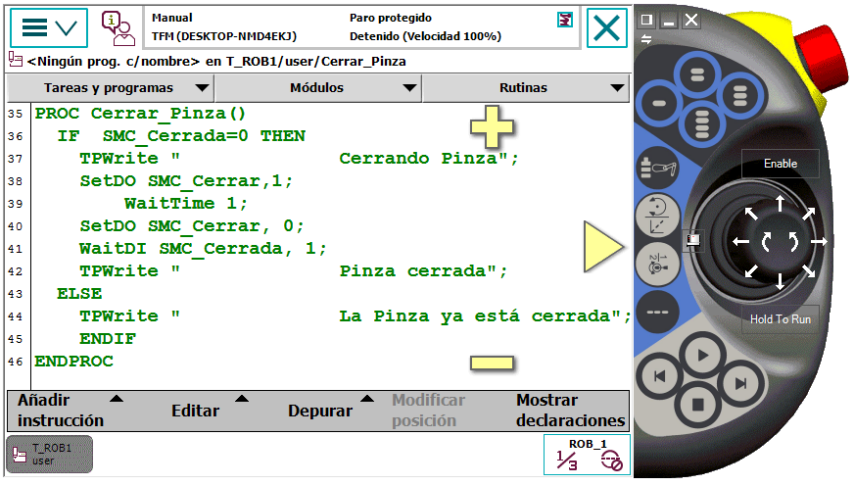
Figura 76: Teclas programables

7.2.5 Funciones Abrir_Pinza() y Cerrar_Pinza() en user

Es necesario sincronizar las funciones Abrir_Pinza y Cerrar_Pinza en el *Editor de Programa, user:*







8 CONCLUSIONES

Tras haber llevado a cabo este trabajo fin de máster, podemos establecer varias conclusiones alcanzadas mediante el mismo.

La primera de ellas que debemos señalar es que el sistema completo ha quedado operativo y listo para uso en la docencia y/o investigación. Por lo que los objetivos planteados al comienzo de este proyecto han sido alcanzados satisfactoriamente.

A nivel formativo, este trabajo ha servido para obtener amplios conocimientos sobre diseño de geometrías e impresión de piezas 3D, diseño y fabricación de PCBs, diseño de circuitos de adaptación electrónica, así como uso, interconexión y programación de robots industriales.

Además, todo ello me ha permitido conocer de primera mano el trabajo sobre instalaciones y conexiones de sistemas robóticos reales, así como enfrentarme directamente la casuística de problemas que en un entorno industrial pueden darse.

9 ACCIONES Y MEJORAS FUTURAS

Una funcionalidad adicional que resultaría interesante implementar es el software de *Integrated Vision* (Visión integrada). La finalidad del sistema *Integrated Vision* de ABB es la de proporcionar un sistema de visión robusto y fácil de usar para aplicaciones de robótica guiada por visión (VGR por sus siglas en inglés) de uso general. El sistema presenta una solución completa de software y hardware que está totalmente integrada con el controlador de robot IRC5 y el entorno de programación *RobotStudio*, el cual está equipado con un entorno de programación de visión con herramientas robustas para localización, inspección e identificación de piezas.

Figura 77: Integrated vision

Para aumentar la versatilidad del robot, siempre podemos adaptar diversas herramientas o elementos terminales que nos permitan llevar a cabo tareas específicas de sujeción, pintura, soldadura, etc…

Figura 78: Elementos terminales

10 REFERENCIAS

[1] ABB Engineering, «DataSheet – IRB 120»

[2] ABB Engineering, «Diagrama de circuitos – IRB 120»

[3] ABB Engineering, « Diagrama de circuitos – IRC5 Compact»

[4] ABB Engineering, «Especificaciones de Producto – IRB 120»

[5] ABB Engineering, « Especificaciones de Producto – IRC5 Compact»

[6] ABB Engineering, « Especificaciones de Producto – Robotware IRC5»

[7] ABB Engineering, «Manual de Producto – IRB 120»

[8] ABB Engineering, «Manual de Producto – IRC5 Compact»

[9] ABB Engineering, «Manual de Referencia Técnica - Instrucciones RAPID»

[10] ABB Engineering, «Manual de Referencia Técnica - Parámetros del sistema»

[11] ABB Engineering, «Manual de Referencia Técnica - RAPID»

[12] ABB Engineering, «Manual del operador - IRC5 con Flexpendant»

[13] ABB Engineering, «Manual del Operador - RobotStudio»

[14] ABB Engineering, «Imágenes, Gráficos y Tablas»

[15] ABB Engineering, «Modelos CAD y bocetos del IRB120»

[16] SMC Corporation, «Actuadores Electricos»

[17] SMC Corporation, «Guia Rápida - LECP6»

[18] SMC Corporation, «Información General - LEHZ25K2-14»

[19] SMC Corporation, «Manual de Instalacion y Mantenimiento - LECP6»

[20] SMC Corporation, «Manual de Instalacion y Mantenimiento - LEHZ25K2-14»

[21] SMC Corporation, «Manual de Operacion - LECP6»

[22] SMC Corporation, «Manual de Operacion - LEHZ25K2-14»

[23] SMC Corporation, «Modelos CAD LEHZ25K2-14»

[24] Arduino, «Imágenes, Tablas y datos técnicos»

[25] Matlab MathWorks, «Getting Started to StateFlow»

[26] Matlab MathWorks, «Simulink Support Package for Arduino Hardware»

[27] CatiaV5 Dassault Systèmes, «Manual CatiaV5 R21»

[28] BQ Prusa, «Imágenes Prusa i3 hephestos»

[29] Isocom Components, «DataSheet ISQ201»

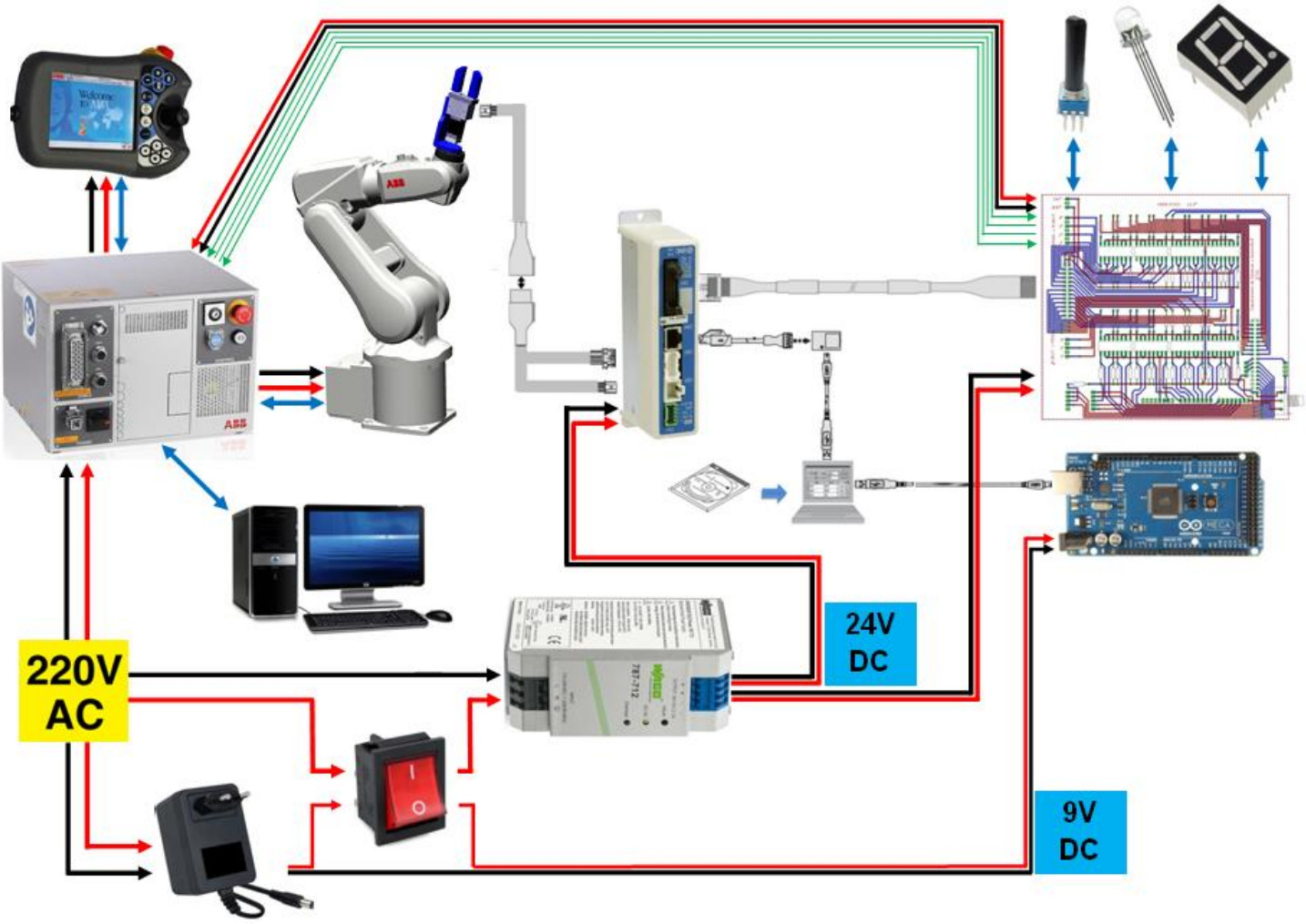
[30] Vishay, «DataSheet CNY74-4H»

[31] reprap.org, «Prusa i3 hephestos»

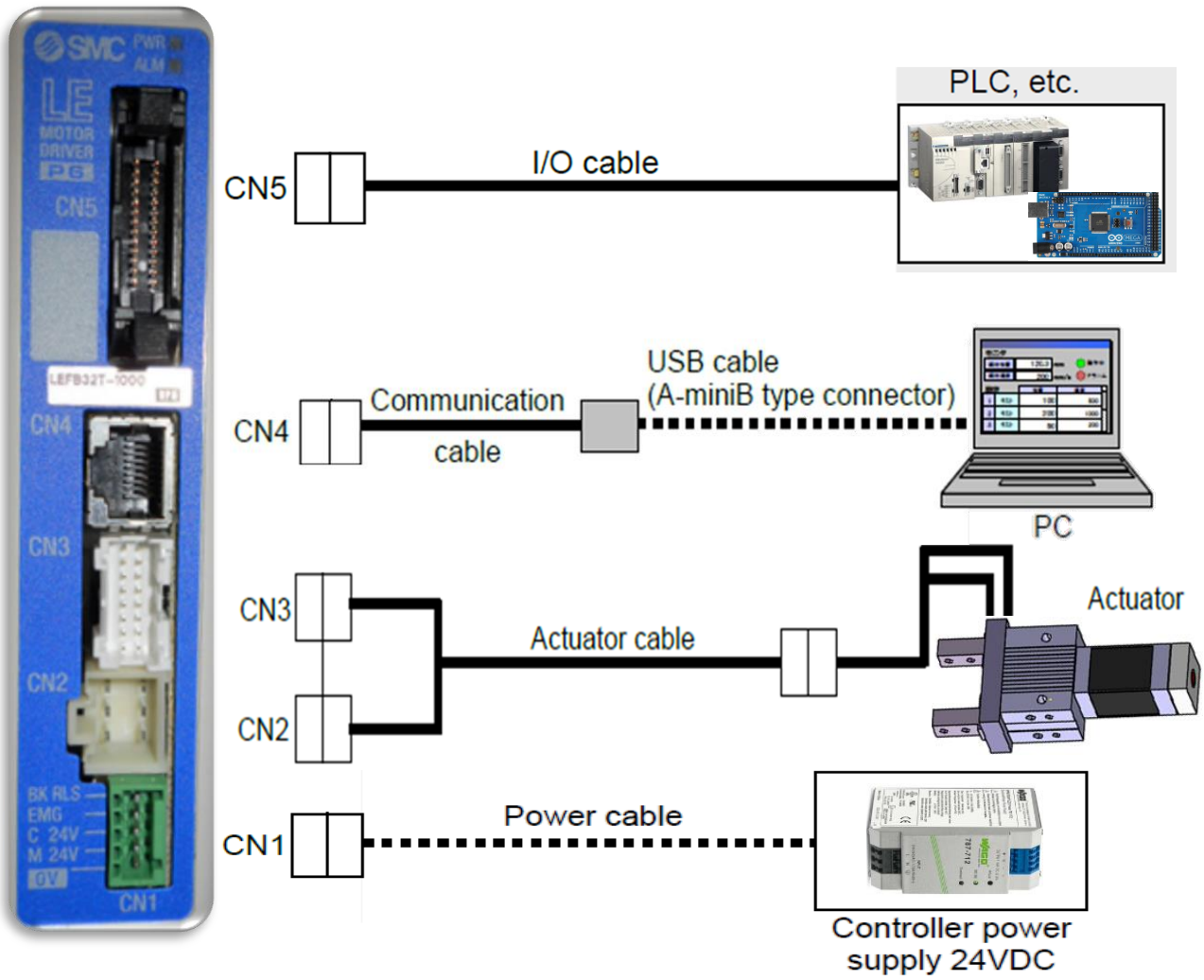
[32] giltesa.com, «Fabricación de circuitos impresos»

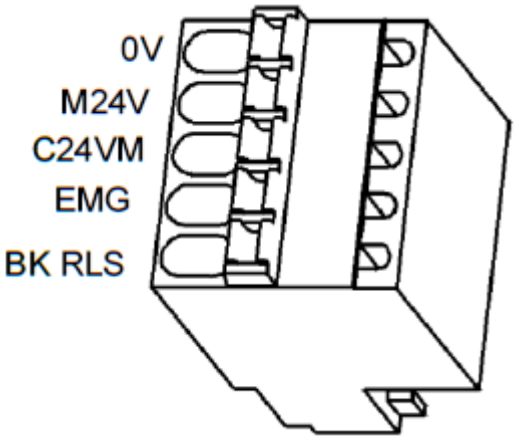
11 ANEXOS

11.1 Anexo A: Esquema general de conexión de los componentes

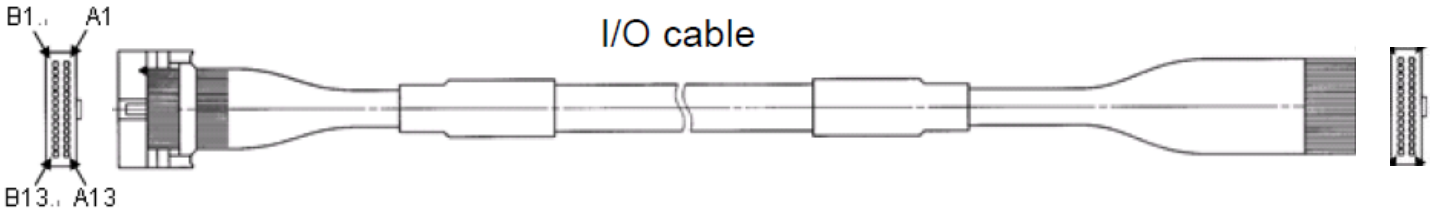


11.2 Anexo B: Diagramas de conexión del controlador LEC-P6





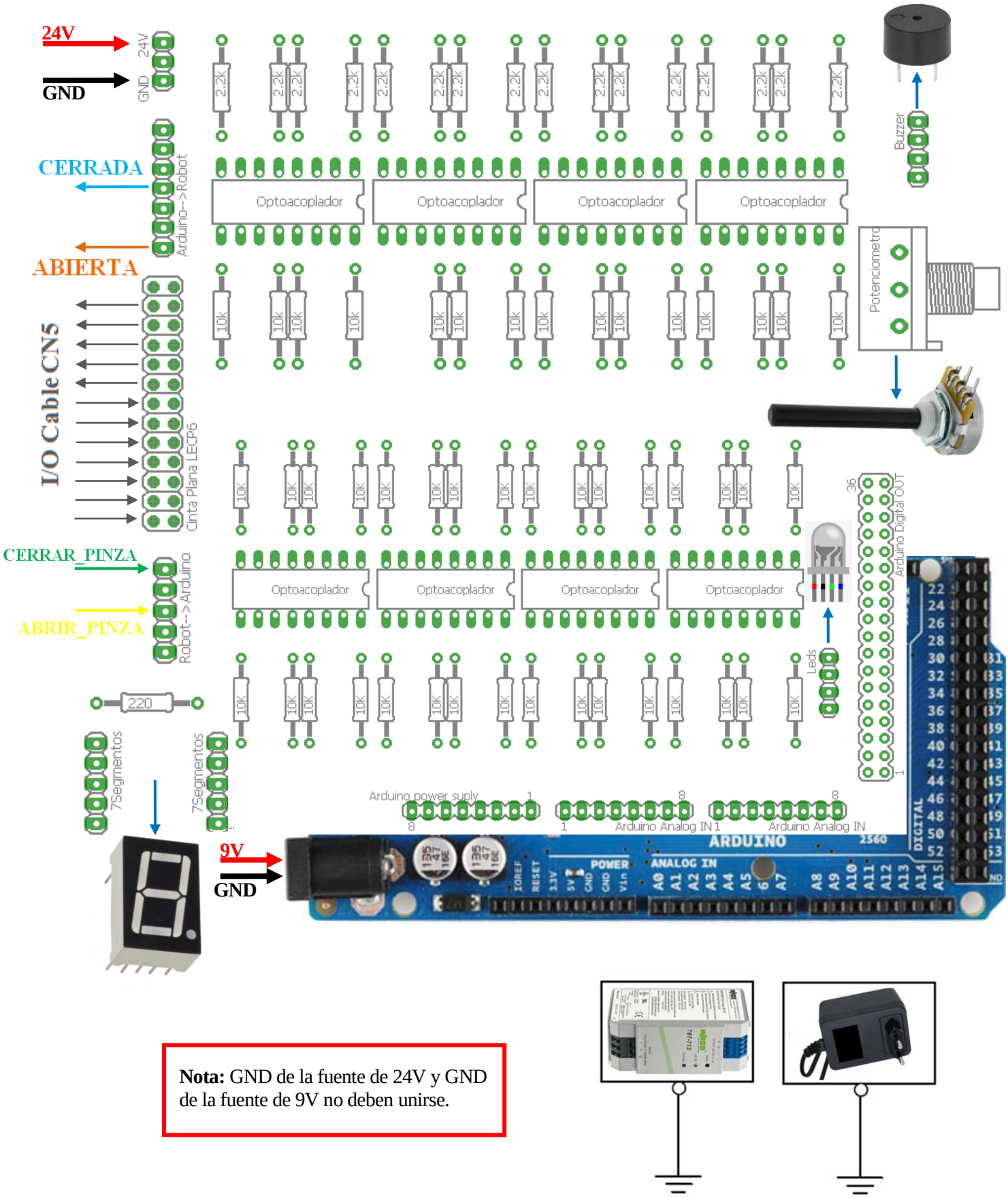
CN1	
TERMINAL	**FUNCIÓN**
0V	Común (-)
M24V	Alimentación Motor (+)
C24VM	Alimentación Control (+)
EMG	Señal de Emergencia (+)
BK RLS	Señal de Bloqueo (+)



CN5 (LEC-P6)	
B1	A1
B2	A2
B3	A3
B4	A4
B5	A5
B6	A6
B7	A7
B8	A8
B9	A9
B10	A10
B11	A11
B12	A12
B13	A13

CN5 (PCB)	
GND	24V
IN1	IN0
IN3	IN2
IN5	IN4
HOLD	SETUP
RESET	DRIVE
OUT0	SVON
OUT2	OUT1
OUT4	OUT3
BUSY	OUT5
SETON	AREA
SVRE	INP
ALARM	ESTOP

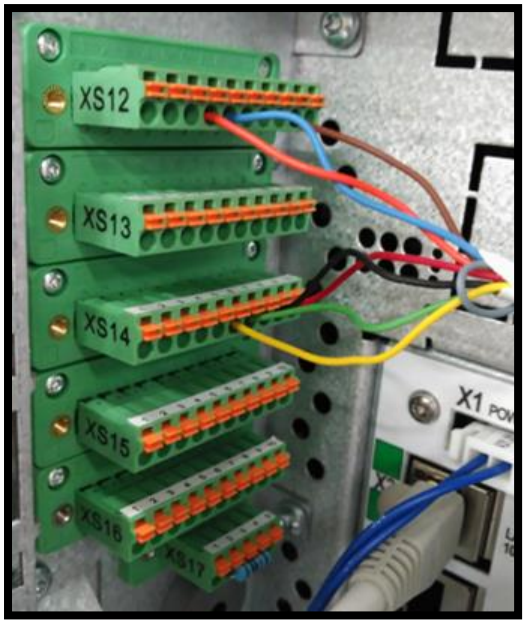
11.3 Anexo C: Diagrama de conexión de la PCB



Nota: GND de la fuente de 24V y GND de la fuente de 9V no deben unirse.

11.4 Anexo D: Diagrama de conexión del controlador IRC5

Borneros de Conexión



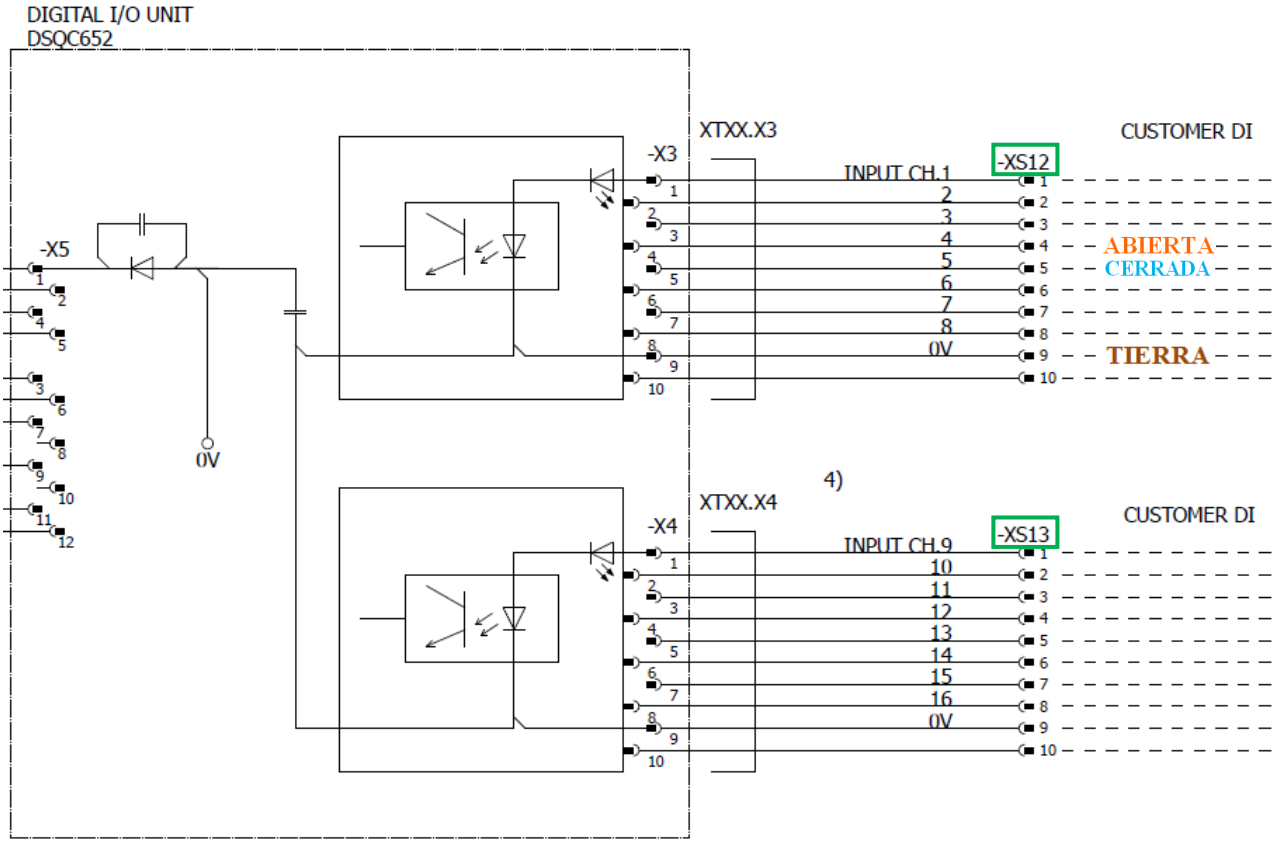
DI

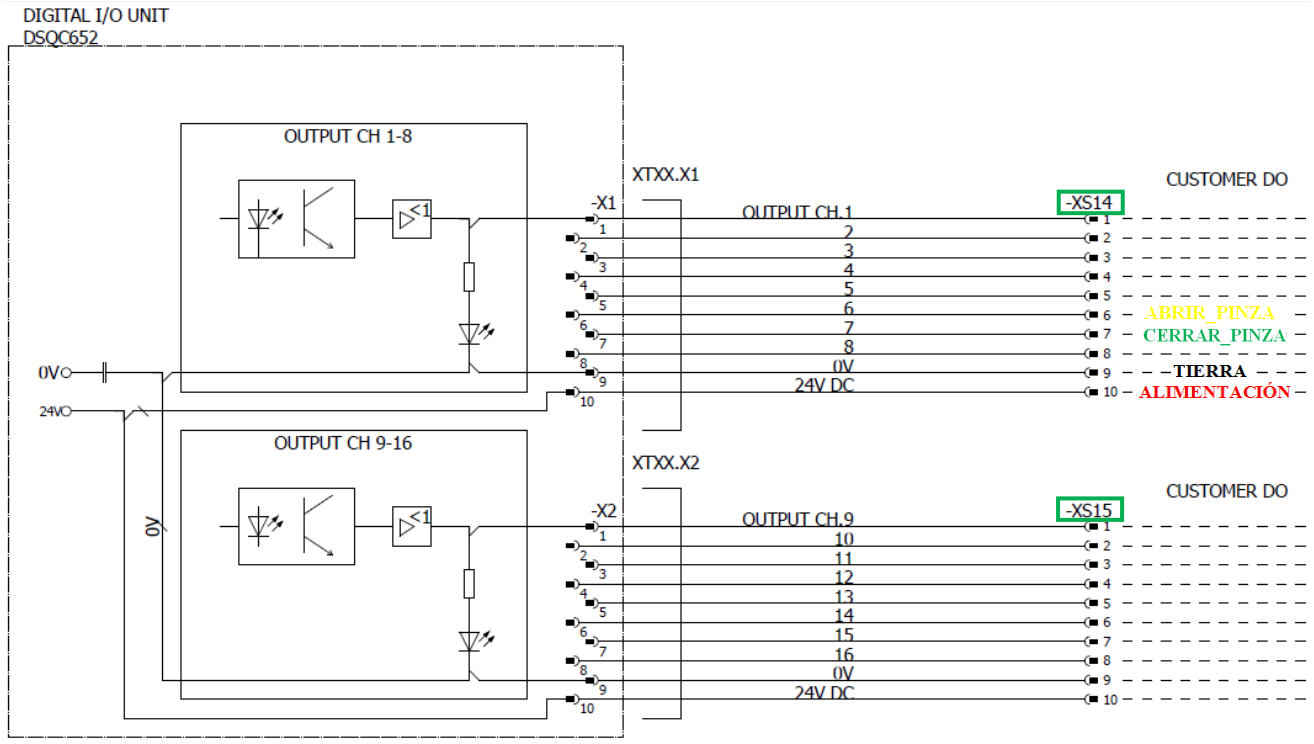
DO

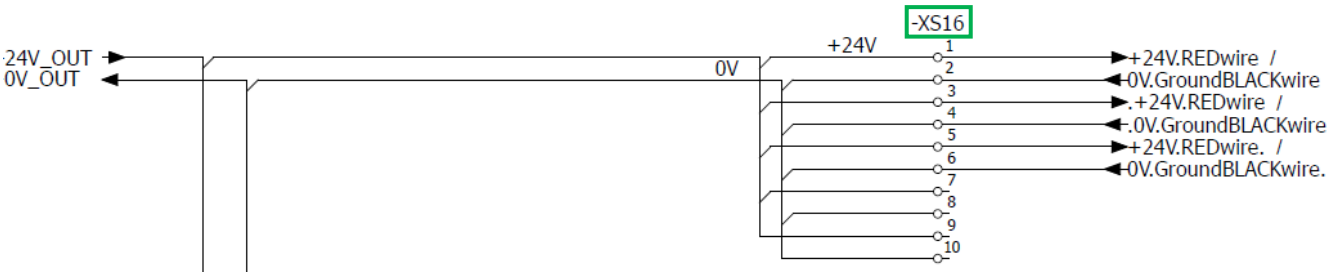
Fuente Interna

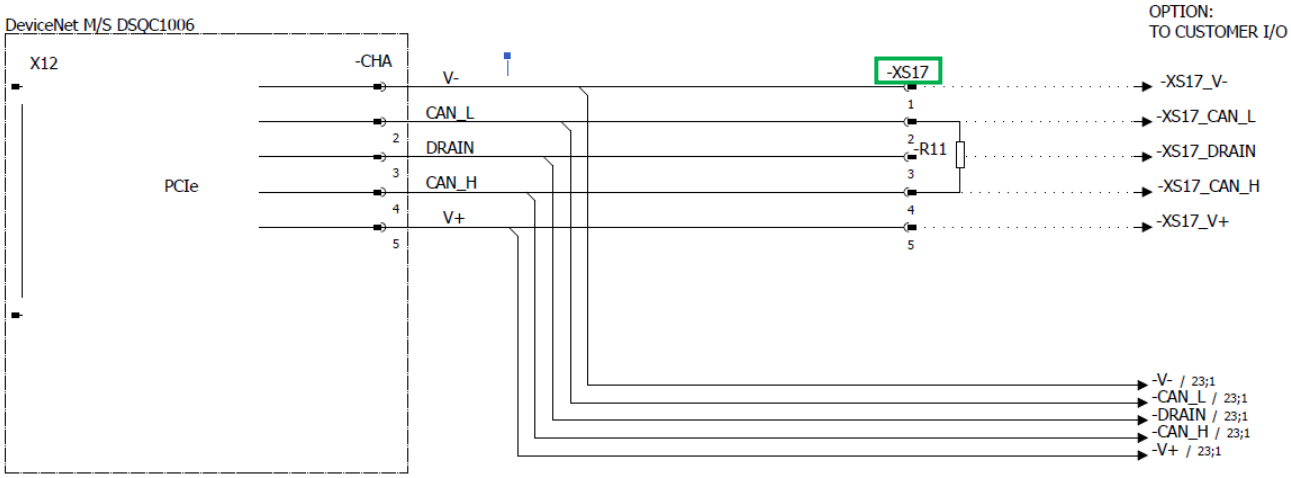
XS12	1	2	3	4	5	6	7	8	0V	NC
XS13	1	2	3	4	5	6	7	8	0V	NC
XS14	1	2	3	4	5	6	7	8	0V	24V
XS15	1	2	3	4	5	6	7	8	0V	24V
XS16	1	2	3	4	5	6	7	8	9	10
XS17				1	2	3	4	5		

Necesario alimentar con 0 y 24V de la Fuente Externa




DIGITAL I/O UNIT
DSQC652
OUTPUT CH 1-8
OUTPUT CH 9-16
XTXX.X1
XTXX.X2
-X1
-X2
0V
24V
OUTPUT CH.1
OUTPUT CH.9
0V
24V DC
CUSTOMER DO
-XS14
-XS15
ABRIR_PINZA
CERRAR_PINZA
TIERRA
ALIMENTACIÓN


24V_OUT
0V_OUT
0V
+24V
-XS16
+24V.REDwire /
-0V.GroundBLACKwire
.+24V.REDwire /
.0V.GroundBLACKwire
+24V.REDwire. /
-0V.GroundBLACKwire.


DeviceNet M/S DSQC1006
X12
PCIe
-CHA
V-
CAN_L
DRAIN
CAN_H
V+
-XS17
R11
OPTION:
TO CUSTOMER I/O
-XS17_V-
-XS17_CAN_L
-XS17_DRAIN
-XS17_CAN_H
-XS17_V+
-V- / 23;1
-CAN_L / 23;1
-DRAIN / 23;1
-CAN_H / 23;1
-V+ / 23;1

11.5 Anexo E: Variables

A continuación se exponen las variables utilizadas en los diversos sistemas.

11.5.1 Controlador LEC-P6

11.5.1.1 Señales del conector CN5 (Cinta Plana)

Señales de Entrada	
COM+	Conecta la alimentación de 24 V para la señal de entrada/salida
COM-	Conecta la alimentación de 0 V para la señal de entrada/salida
IN0 – IN5	Nº bits especificado en los datos de paso (la entrada se define en la combinación de IN0 a 5)
SETUP	Instrucción para retorno al origen
HOLD	El funcionamiento se detiene temporalmente
DRIVE	Instrucción para accionamiento
RESET	Reinicio de alarma e interrupción del funcionamiento
SVON	Instrucción de activación del servoaccionamiento

Señales de Salida	
OUT0 – OUT5	Salidas del nº de datos de paso durante el funcionamiento
BUSY	Salidas cuando el actuador está en movimiento
AREA	Salidas dentro del rango de ajuste de salida del área de datos de paso
SETON	Salidas durante el retorno al origen
INP	Salidas cuando se alcanza la posición de destino o la fuerza objetivo (Se activa cuando se completa el posicionamiento o el empuje.)
SVRE	Salida cuando el servoaccionamiento está activado
***ESTOP**	No hay salida cuando se ordena la parada EMG
***ALARM**	No hay salida cuando se genera la alarma

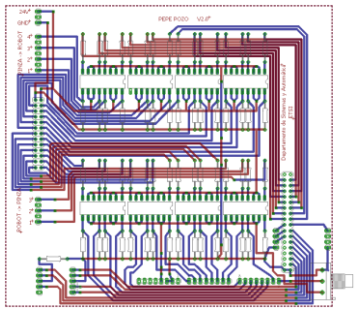
11.5.1.2 Datos de Paso (Step Data)

Para configurar el LEC-P6 es necesario conectar el cable CN4 e instalar el software *ACT Controller*.

No.	Move	Speed mm/s	Position mm	Accel mm/s^2	Decel mm/s^2	PushingF %	TriggerLV %	PushingSp mm/s	Moving F %	Area1 mm	Area2 mm	In pos mm
0	ABS	100	20.00	1000	1000	0	0	0	100	18.00	22.50	0.5
1	ABS	50	10.00	1000	1000	70	60	5	100	6.0	12.0	1.5

Nº	Número de la posición a configurar
Modo de Movimiento	Cuando se requiera la posición absoluta, confi gurar en "Absoluto". Cuando se requiera la posición relativa, confi gurar en "Relativo".
Velocidad	Velocidad de traslado hasta la posición de destino.
Posición	Posición de destino
Aceleración	Parámetro que defi ne la rapidez con la que el actuador alcanza la velocidad de ajuste. Cuanto mayor es el valor de ajuste, más rápido se alcanzará la velocidad de ajuste.
Deceleración	Parámetro que defi ne la rapidez con la que el actuador se detiene. Cuanto mayor es el valor de ajuste, más rápido se detiene.
Fuerza de Empuje	Se defi ne el factor de fuerza de empuje.
Disparador LV	Condición que activa la señal de salida INP. La señal de salida INP se activa cuando la fuerza generada supera el valor. El nivel de activación debe ser la fuerza de empuje o inferior.
Velocidad de Empuje	Velocidad de empuje durante el empuje. Si la velocidad de ajuste es elevada, el actuador eléctrico y las piezas de trabajo pueden resultar dañadas debido al impacto de las mismas contra el extremo, por lo que el valor de la velocidad debe ser más bajo.
Fuerza de Desplazamiento	Par máximo durante la operación de posicionamiento
Área1 – Área 2	Condición que activa la señal de salida AREA
Posición de Entrada	Distancia de traslado durante el empuje. Si la distancia de traslado supera el valor de ajuste, el producto se detiene, incluso si no se encuentra en una operación de empuje. Si se supera la distancia de traslado, la señal de salida INP no se activará.

11.5.2 Placa PCB



11.5.2.1 Eagle PCB

Las siguientes señales se utilizan en el diseño de la placa son el software *Eagle PCB.*

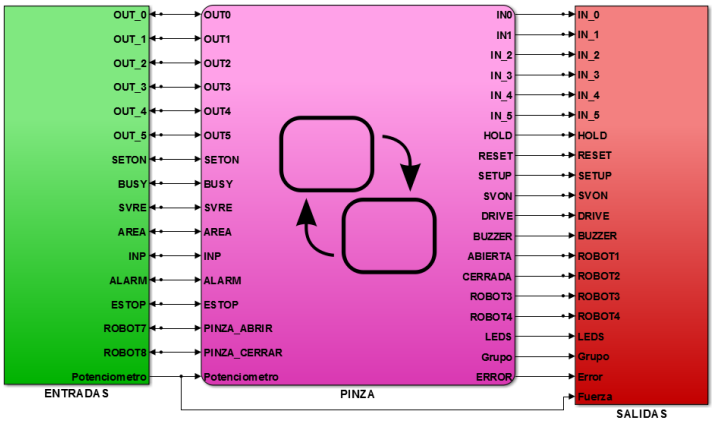
3.3V	Conexión de Arduino a 3.3V
5V	Conexión de Arduino a 5V
VCC	Alimentación a 24V
GND	Tierra de la fuente de alimentación de 24V
GND_ARDU	Tierra de Arduino
A0 – A15	Entradas analógicas Arduino (0-5V)
ARDU_22 – ARDU_53	Salidas digitales Arduino (0-5V)
ROBOT1 – ROBOT4	Salidas digitales al controlador IRC5 (0-24V)
ROBOT6 – ROBOT8	Entradas digitales del controlador IRC5 (0-24V)

IN0	IN1	OUT0	OUT1
IN2	IN3	OUT2	OUT3
IN4	IN5	OUT4	OUT5
HOLD	SETUP	BUSY	SETON
RESET	DRIVE	AREA	SVRE
SVON	INP	ALARM	STOP

11.5.2.2 Matlab – Arduino

Las siguientes señales se utilizan en la programación de Arduino Mega con *Matlab 2013b.*

ENTRADAS	PINZA		SALIDAS
OUT_0	OUT0	IN0	IN_0
OUT_1	OUT1	IN1	IN_1
OUT_2	OUT2	IN2	IN_2
OUT_3	OUT3	IN3	IN_3
OUT_4	OUT4	IN4	IN_4
OUT_5	OUT5	IN5	IN_5
SETON	SETON	HOLD	HOLD
BUSY	BUSY	RESET	RESET
SVRE	SVRE	SETUP	SETUP
AREA	AREA	SVON	SVON
INP	INP	DRIVE	DRIVE
ALARM	ALARM	BUZZER	BUZZER
ESTOP	ESTOP	ABIERTA	ROBOT1
ROBOT7	PINZA_ABRIR	CERRADA	ROBOT2
ROBOT8	PINZA_CERRAR	LEDS	LEDS
Potenciometro	Potenciometro	Grupo	Grupo
		Error	Error
			Fuerza



11.5.3 IRC5

Las siguientes señales se utilizan en la programación del controlador del robot IRC5

Nombre	Tipo de señal	Device	Device Mapping
SMC_Abierta	Digital Imput	D652_10	6
SMC_Cerrada	Digital Imput	D652_10	4
SMC_Abrir	Digital Output	D652_10	5
SMC_Cerrar	Digital Output	D652_10	3

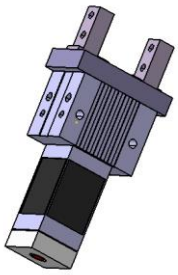
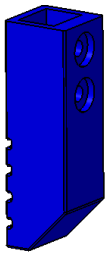
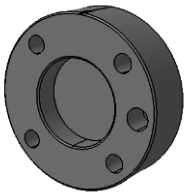
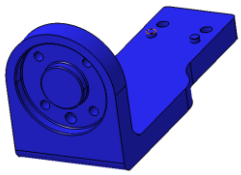
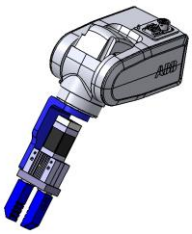
Borneros de Conexión

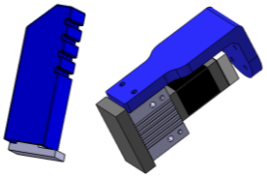
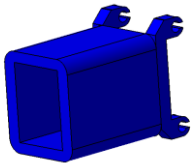
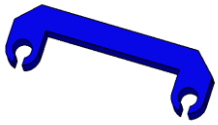
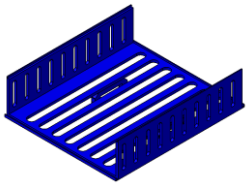
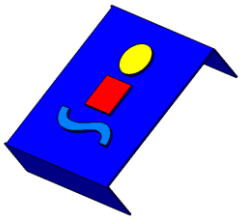
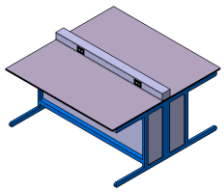
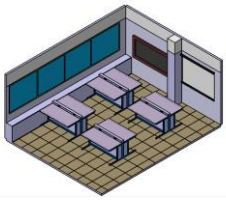
DI	**XS12**	1	2	3	4	5	6	7	8	0V	NC
	XS13	1	2	3	4	5	6	7	8	0V	NC
DO	**XS14**	1	2	3	4	5	6	7	8	0V	24V
	XS15	1	2	3	4	5	6	7	8	0V	24V
Fuente Interna	**XS16**	1	2	3	4	5	6	7	8	9	10
	XS17				1	2	3	4	5		

Necesario alimentar con 0 y 24V de la Fuente Externa

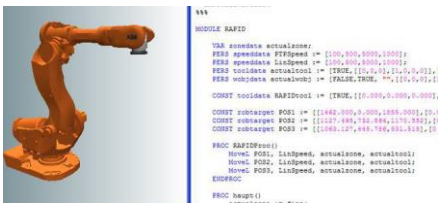
11.6 Anexo F: Componentes modelados en CATIA V5

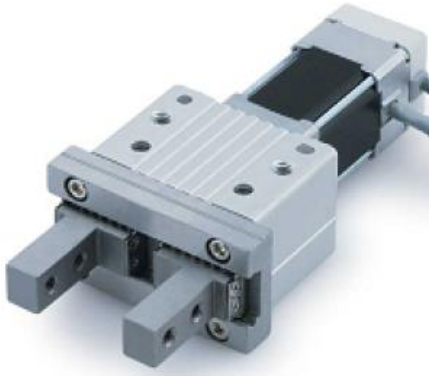
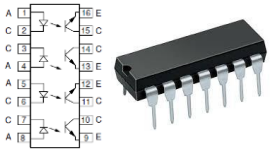
A continuación se presenta una tabla con todos los componentes modelados en Catia V5, la mayoría de ellos impresos en 3D. Para acceder a ellos solo hay que hacer click sobre el nombre del archivo que se desea abrir. Es necesario tener instalados los programas pertinentes para la apertura de dicho enlace.

NOMBRE	IMAGEN	PDF	CATIA
Pinza		Pinza	PINZA
Dedo de la Pinza		Dedo de la pinza	DEDOS
Muñeca del Robot		Muñeca del Robot	LINK6
Pieza de Unión		Pieza de union	PIEZA
Unión Muñeca - Robot - Pinza		Unión Muñeca - Robot - Pinza	PINZA_conjunto

Mecanismos para RobotStudio		Mecanismo para RobotStudio	DEDO_mec PINZA_mec
Soporte para Interruptor		Soporte para interruptor	INTERRUPTOR
Sujeción a carril DIN		Sujecion a carril DIN	CARRIL_DIN
Caja PCB		Caja PCB	CAJA
Tapa PCB		Tapa PCB	TAPA
Mesa de Trabajo		Mesa de trabajo	MESA
Laboratorio		Laboratorio	LABORATORIO
Sujeción cable 1		Cable1	CABLE1
Sujeción cable 2		Cable2	CABLE2

11.7 Anexo G: Manuales de operación, productos y datasheets

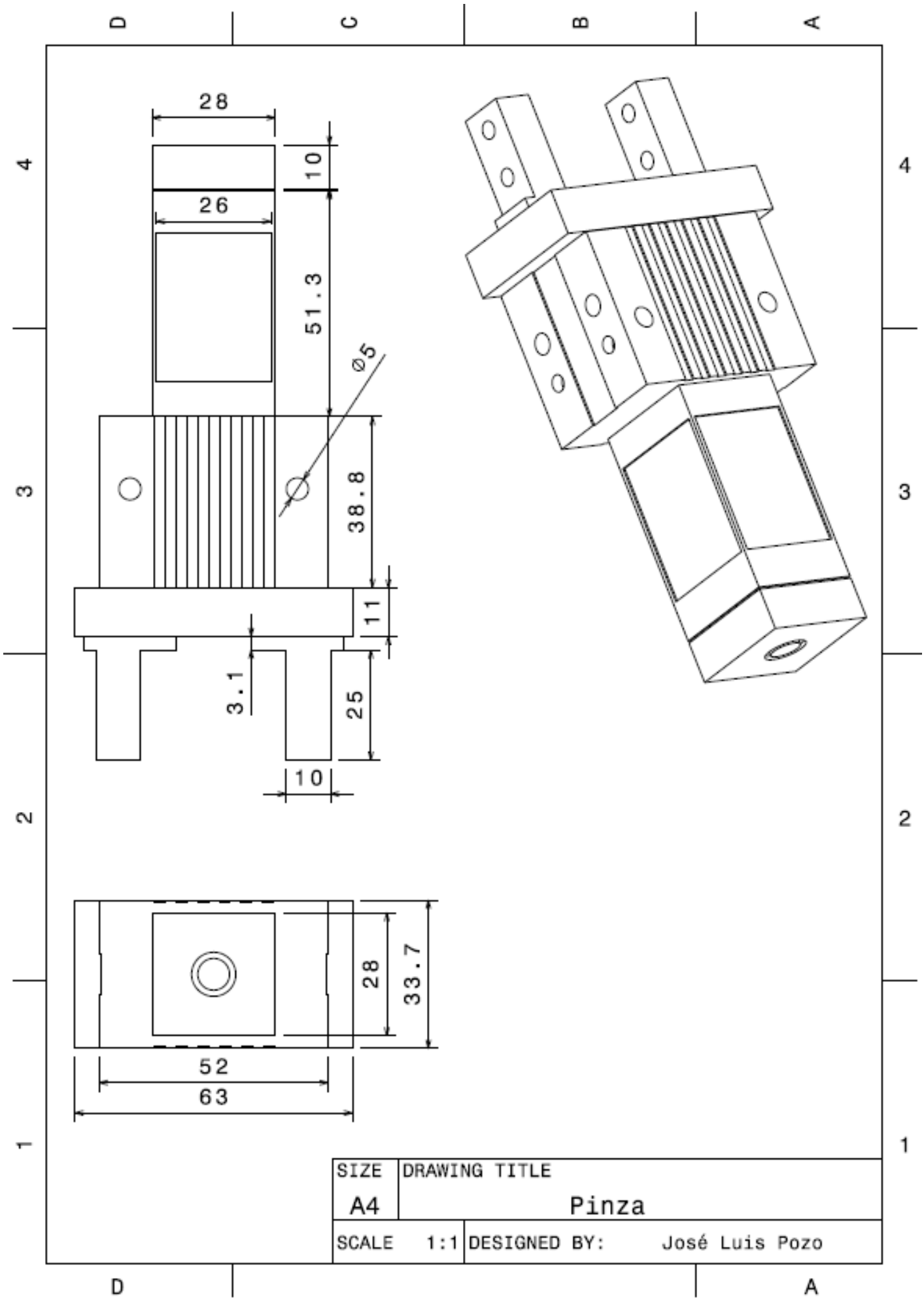
PRODUCTO	PDF
	DataSheet - IRB 120
	Especificaciones de Producto - IRB 120
	Manual de Producto - IRB 120
	Diagramas de Circuitos - IRB 120
	Especificaciones de Producto - IRC5 Compact
	Especificaciones de Producto - RobotWare IRC5
	Manual de Producto - IRC5 Compact
	Diagramas de Circuitos - IRC5 Compact
	Manual de Referencia Técnica - Instrucciones RAPID
	Manual de Referencia Técnica - Parámetros del sistema
	Manual de Referencia Técnica - RAPID
	Manual del Operador - RobotStudio

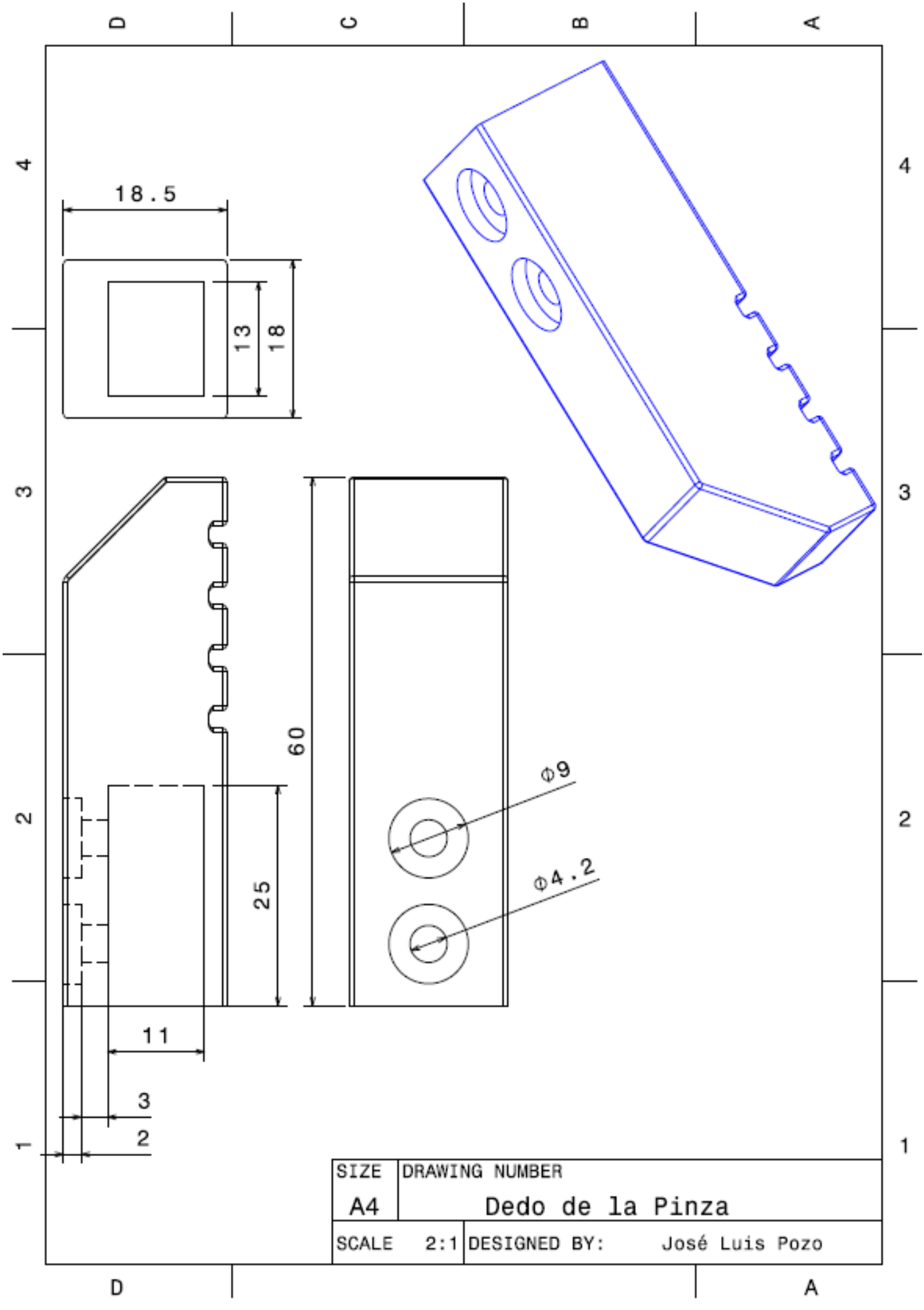
	Manual del operador - IRC5 con Flexpendant
	Guia Rápida - LECP6
	Manual de Instalacion y Mantenimiento - LECP6
	Manual de Operacion - LECP6
	Actuadores Electricos
	Información General - Pinza
	Manual de Instalacion y Mantenimiento - Pinza
	Manual de Operacion - Pinza
	DataSheet - Fuente de Alimentación
	DataSheet - Arduino Mega 2560
	ISQ-201 CNY74-4H

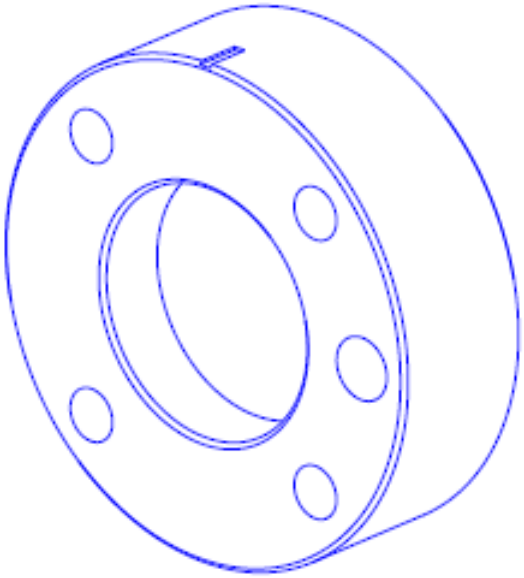
11.8 Anexo H: Planos de los componentes modelados en CATIA V5

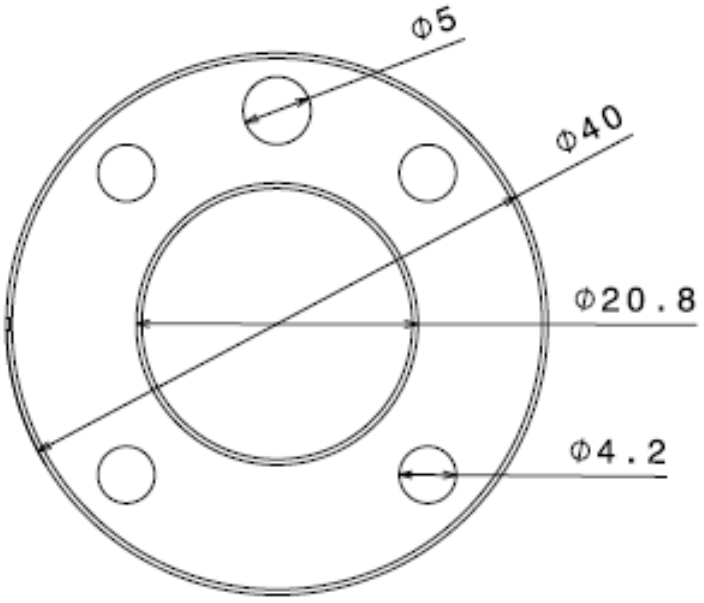
A continuación se adjuntan todos los planos de los componentes utilizados.

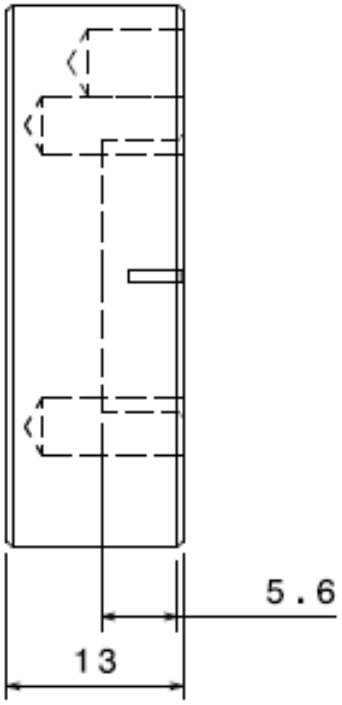
NOMBRE	NÚMERO
Pinza	1
Dedo de la Pinza	2
Muñeca del Robot	3
Pieza de Unión	4
Unión Muñeca - Robot - Pinza	5
Mecanismos para RobotStudio	6
Soporte para Interruptor	7
Sujeción a carril DIN	8
Sujeccion a carril DIN 2	9
Caja PCB	10
Tapa PCB	11
Mesa de Trabajo	12
Laboratorio	13


28
10
26
51.3
Ø5
38.8
11
3.1
25
10
28
33.7
52
63
SIZE
A4
DRAWING TITLE
Pinza
SCALE 1:1
DESIGNED BY: José Luis Pozo

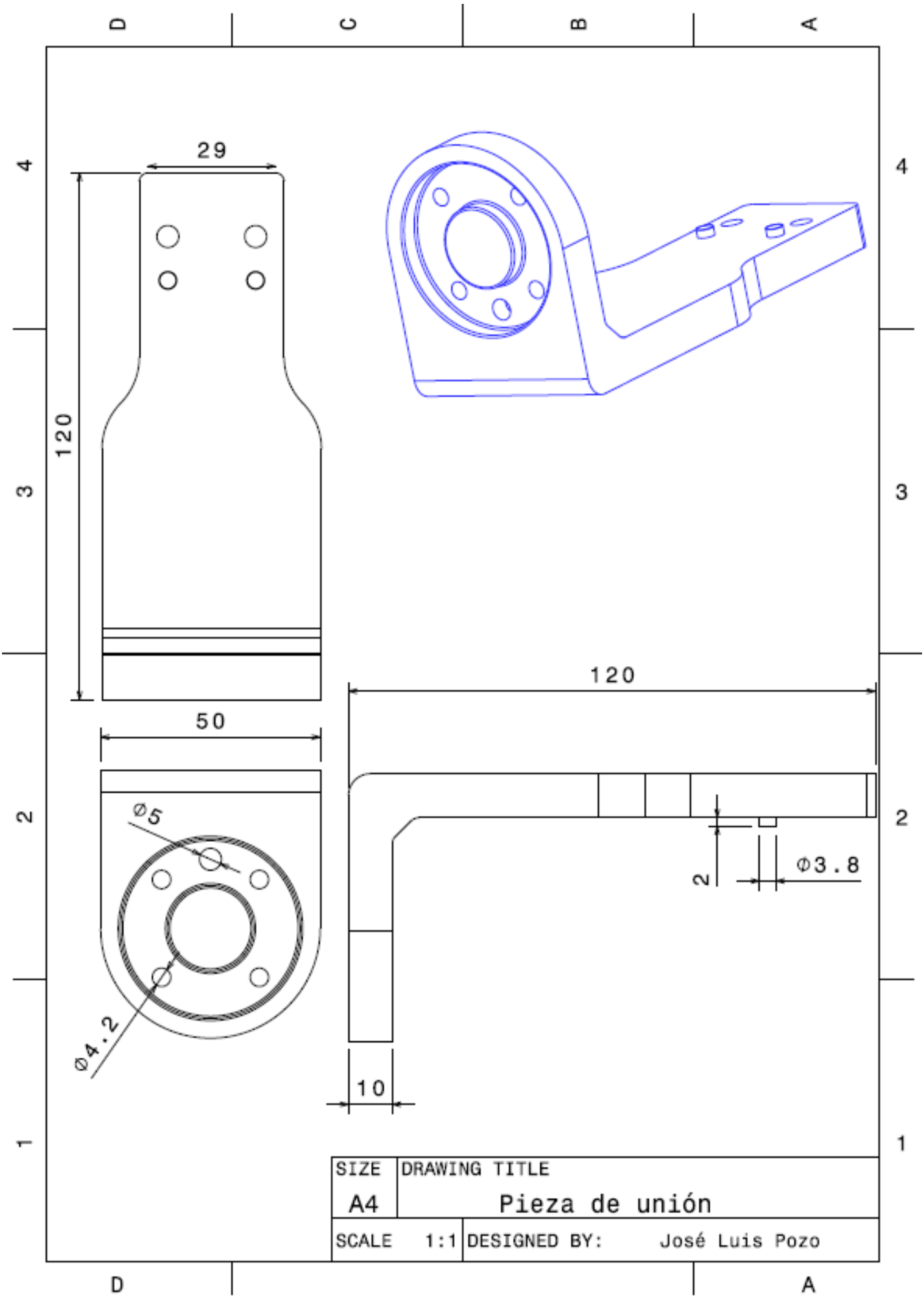

D
C
B
A
4
3
2
1
18.5
13
18
60
25
11
3
2
Φ9
Φ4.2
SIZE
A4
DRAWING NUMBER
Dedo de la Pinza
SCALE 2:1
DESIGNED BY: José Luis Pozo

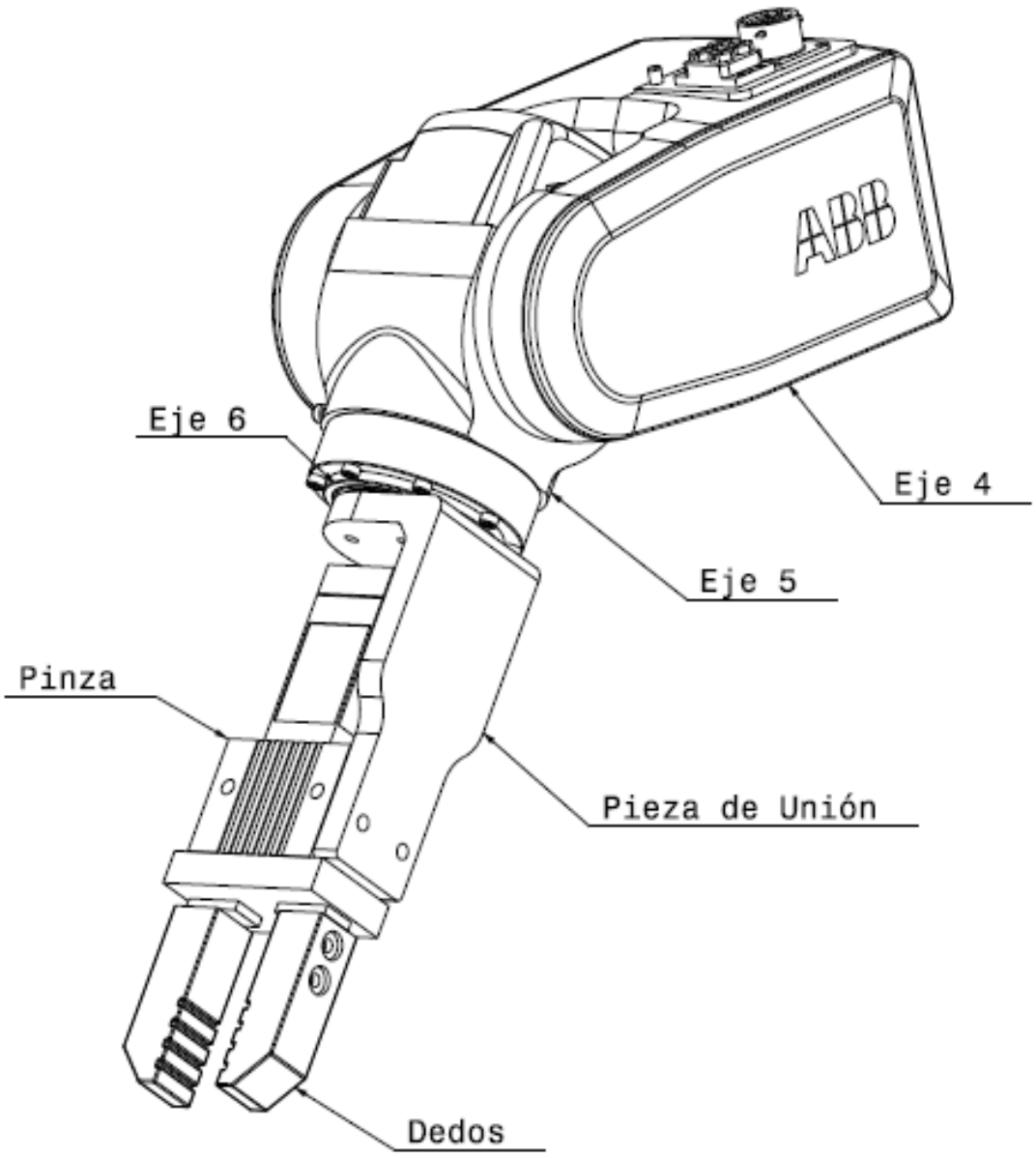




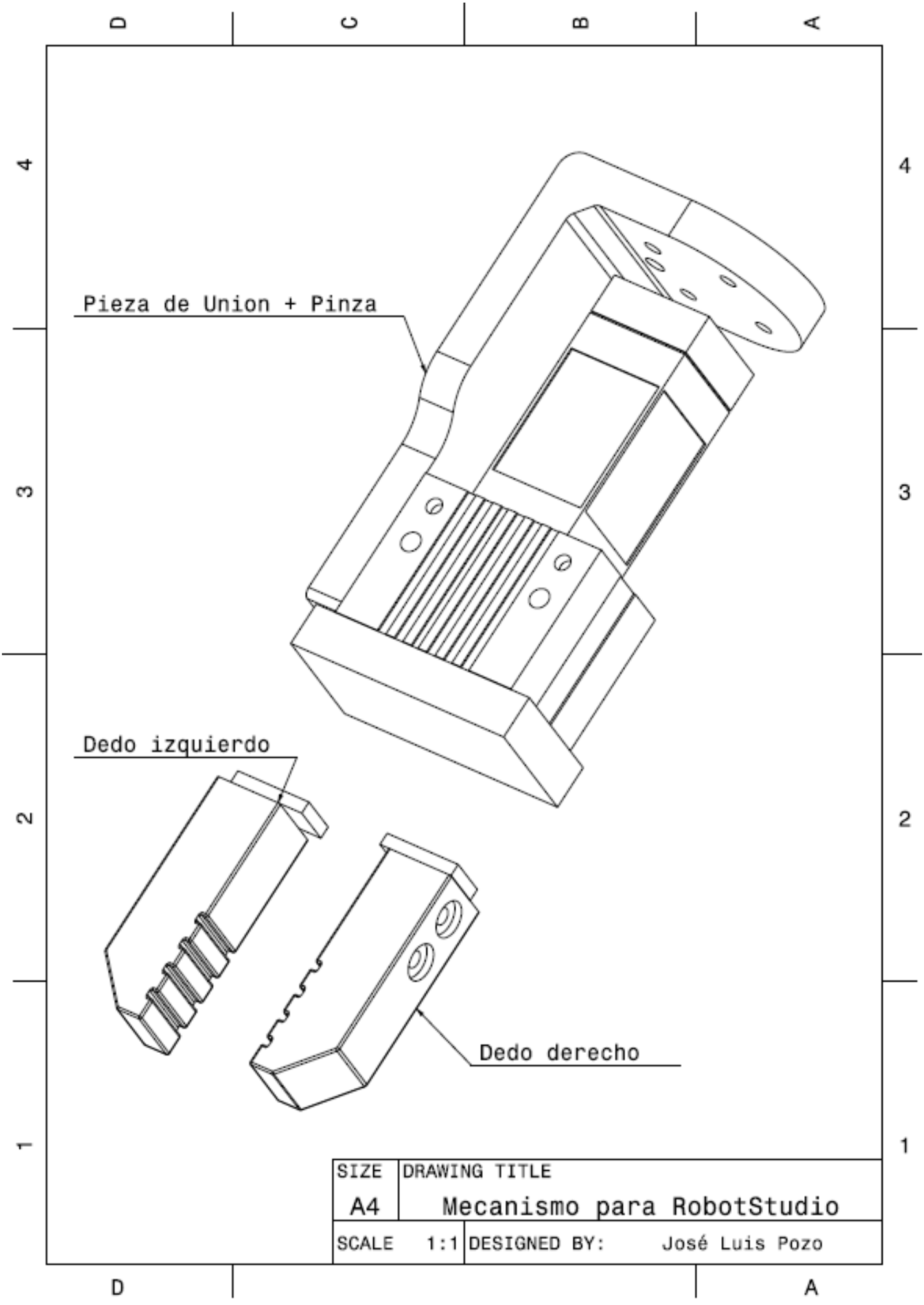


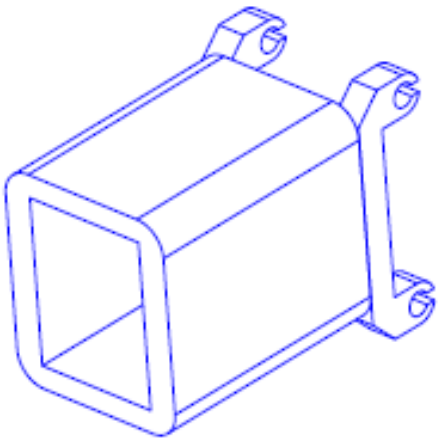
SIZE	DRAWING TITLE
A4	Muñeca del Robot
SCALE 2:1	DESIGNED BY: José Luis Pozo

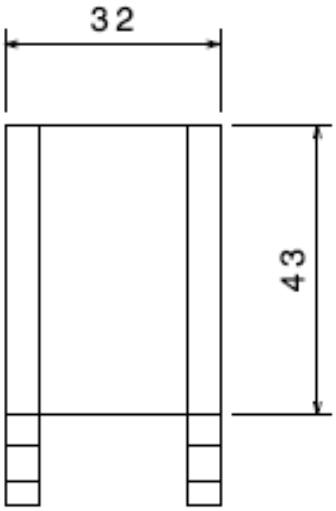

D
C
B
A
4
3
2
1
29
120
50
120
Ø5
Ø4.2
10
2
Ø3.8
SIZE
A4
DRAWING TITLE
Pieza de unión
SCALE 1:1
DESIGNED BY: José Luis Pozo

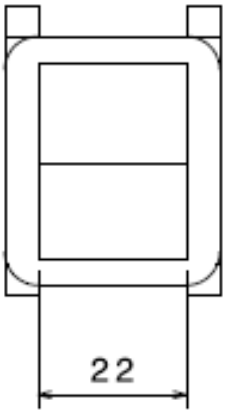


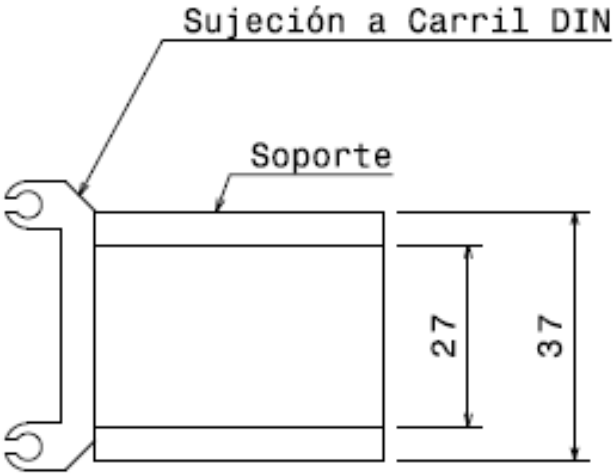
SIZE	DRAWING TITLE
A4	Unión Muñeca Robot - Pinza
SCALE 1:2	DESIGNED BY: José Luis Pozo


D
C
B
A
4
3
2
1
Pieza de Union + Pinza
Dedo izquierdo
Dedo derecho
SIZE
A4
DRAWING TITLE
Mecanismo para RobotStudio
SCALE 1:1
DESIGNED BY: José Luis Pozo

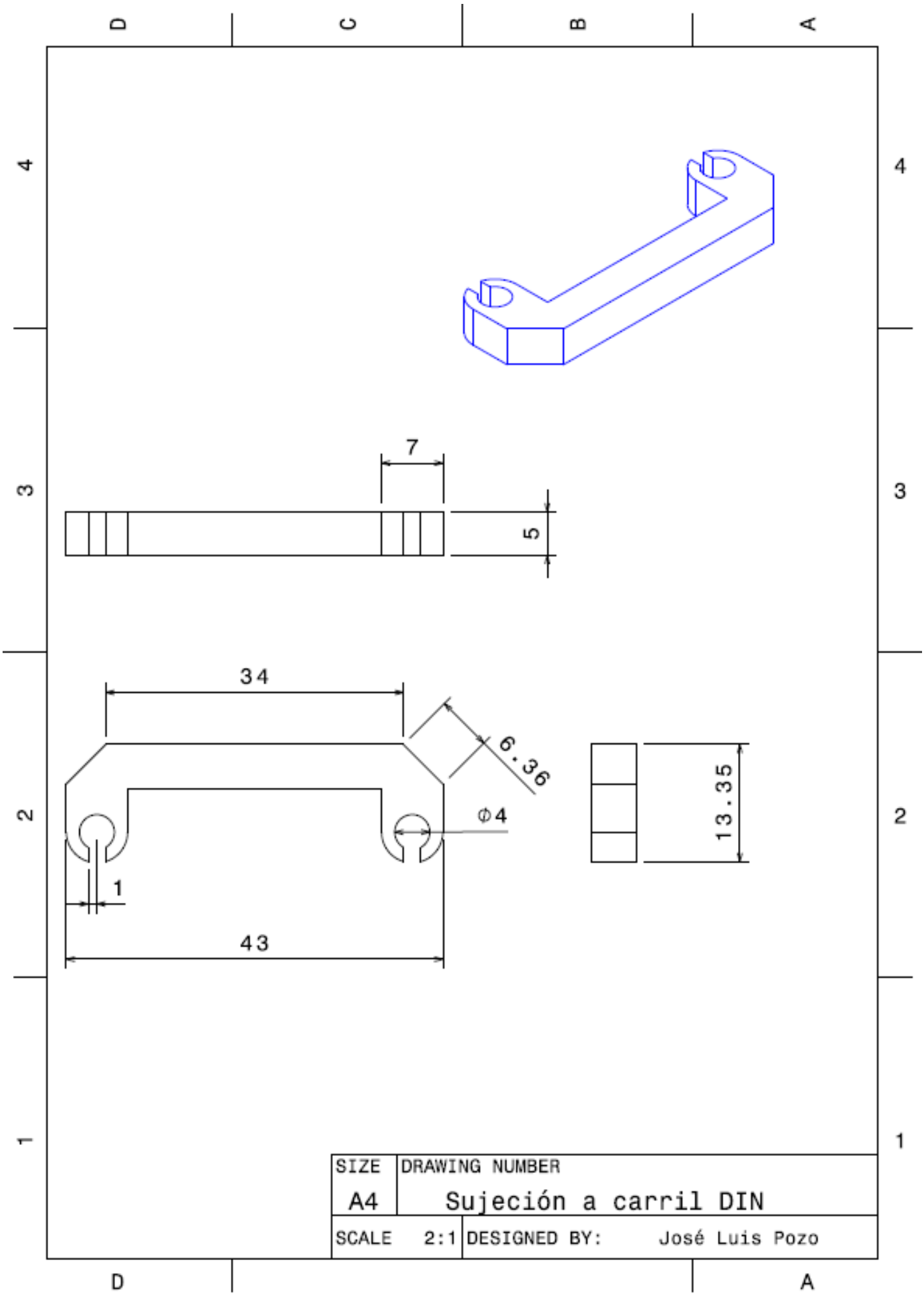


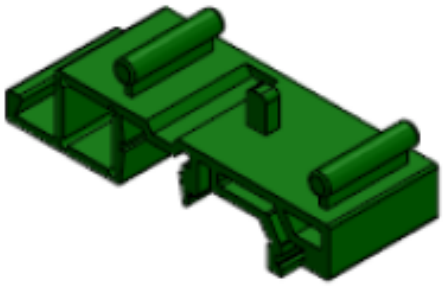


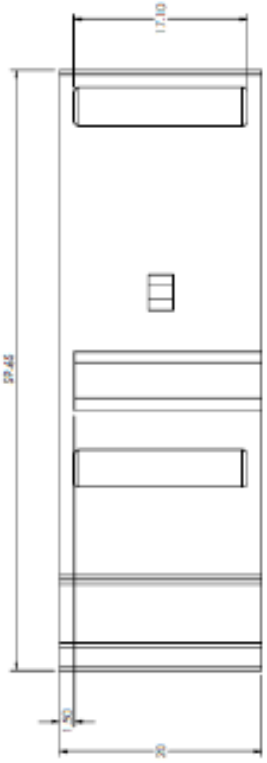


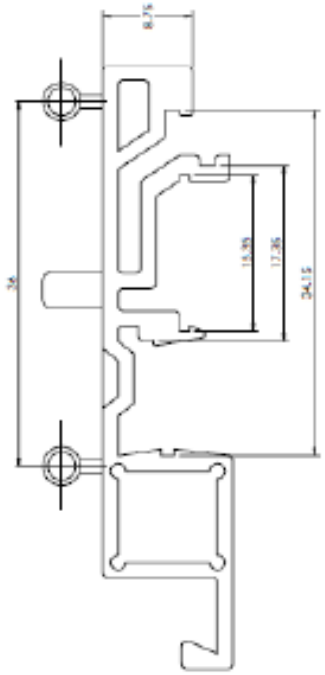


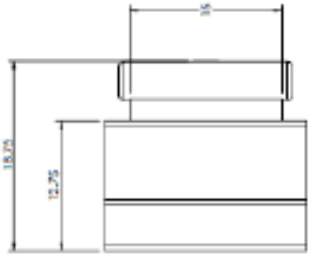
SIZE	DRAWING TITLE
A4	Soporte para interruptor
SCALE 1:1	DESIGNED BY: José Luis Pozo


D
C
B
A
4
3
2
1
7
5
34
6.36
Φ4
1
43
13.35
SIZE
A4
DRAWING NUMBER
Sujeción a carril DIN
SCALE 2:1
DESIGNED BY: José Luis Pozo

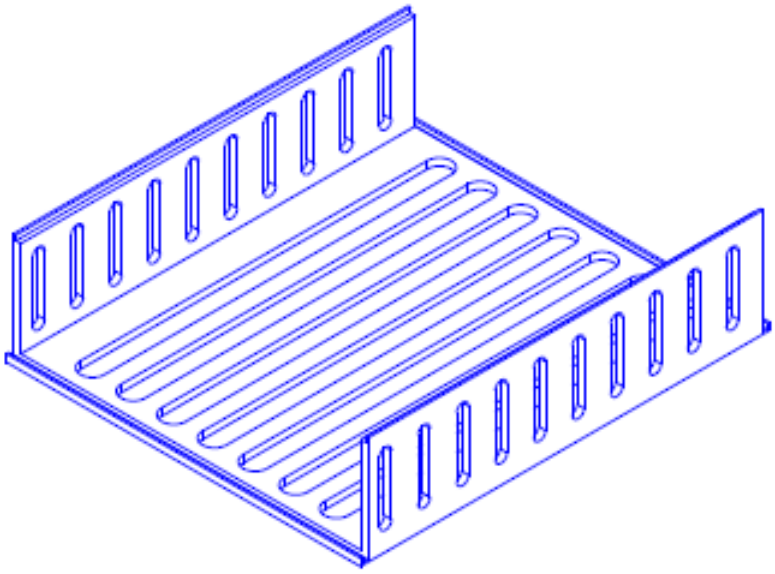


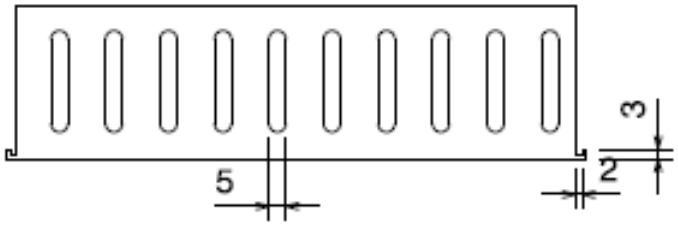


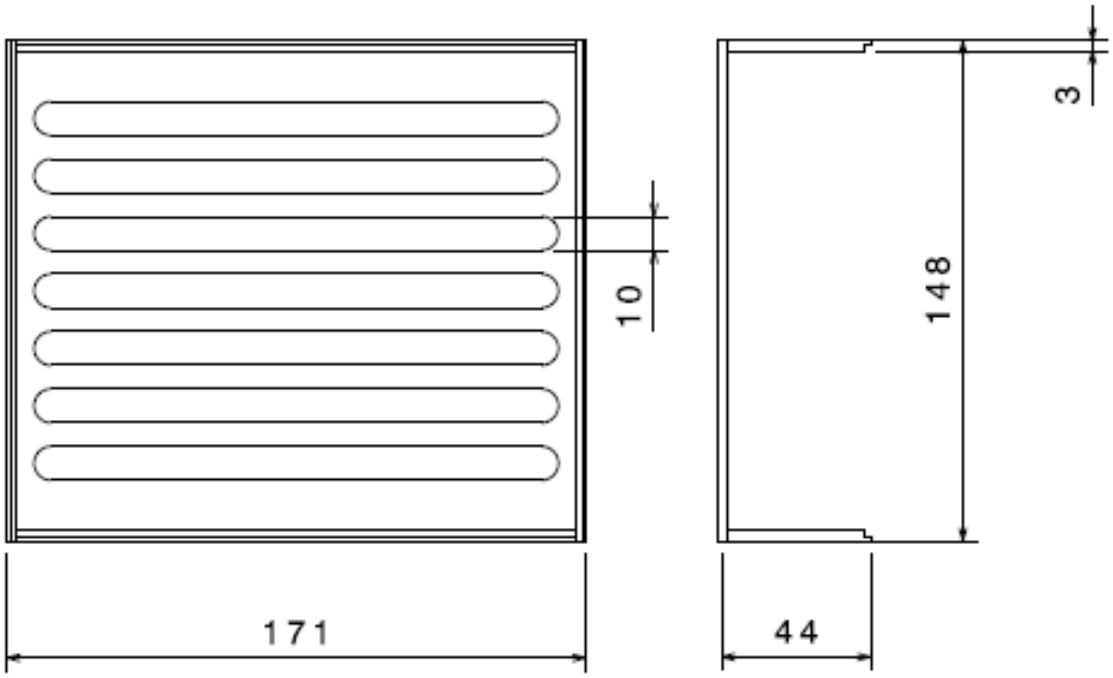




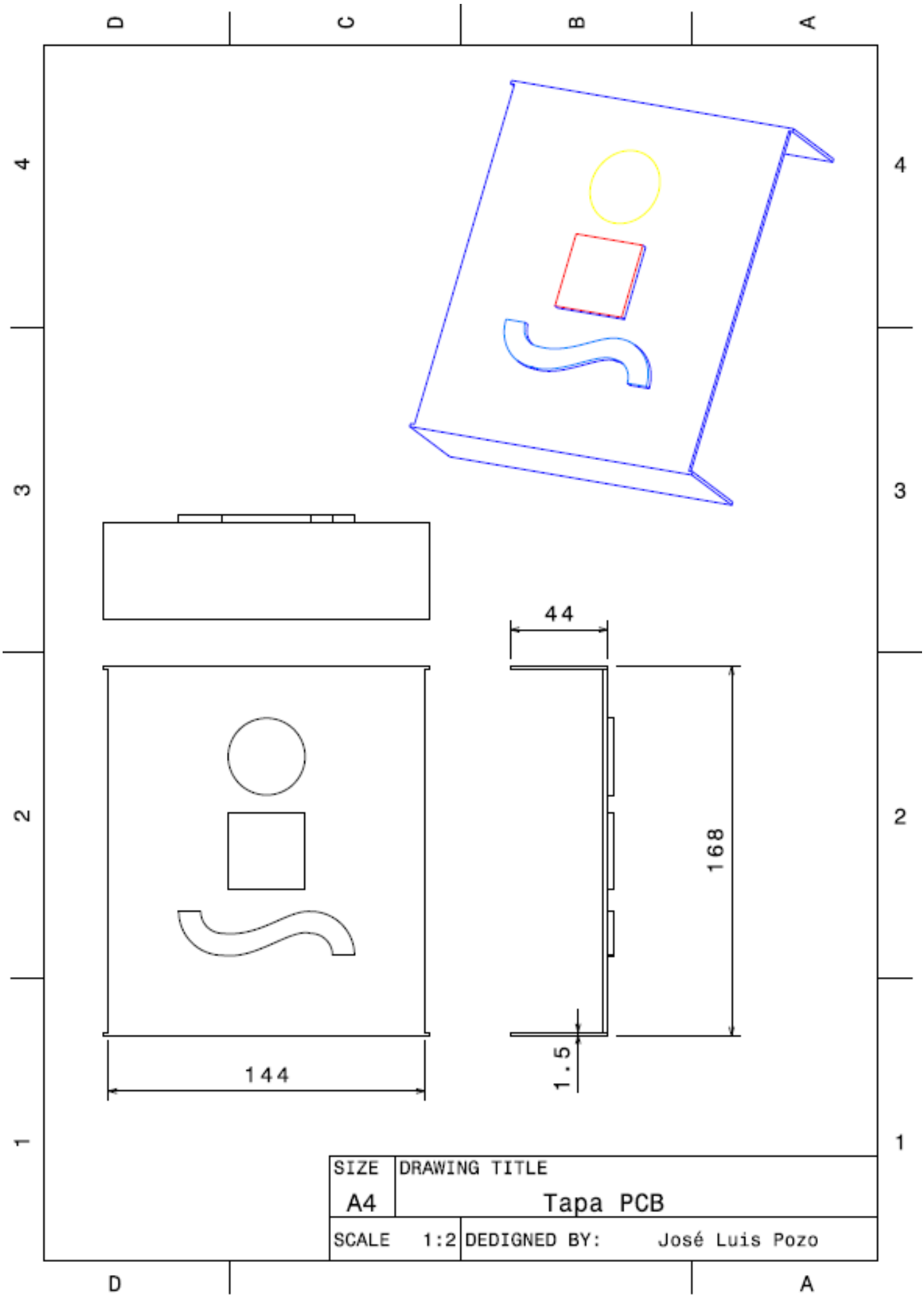
SIZE	DRAWING TITLE
A4	Sujección a carril DIN
SCALE 1:1	DESIGNED BY: CAMDENBOSS

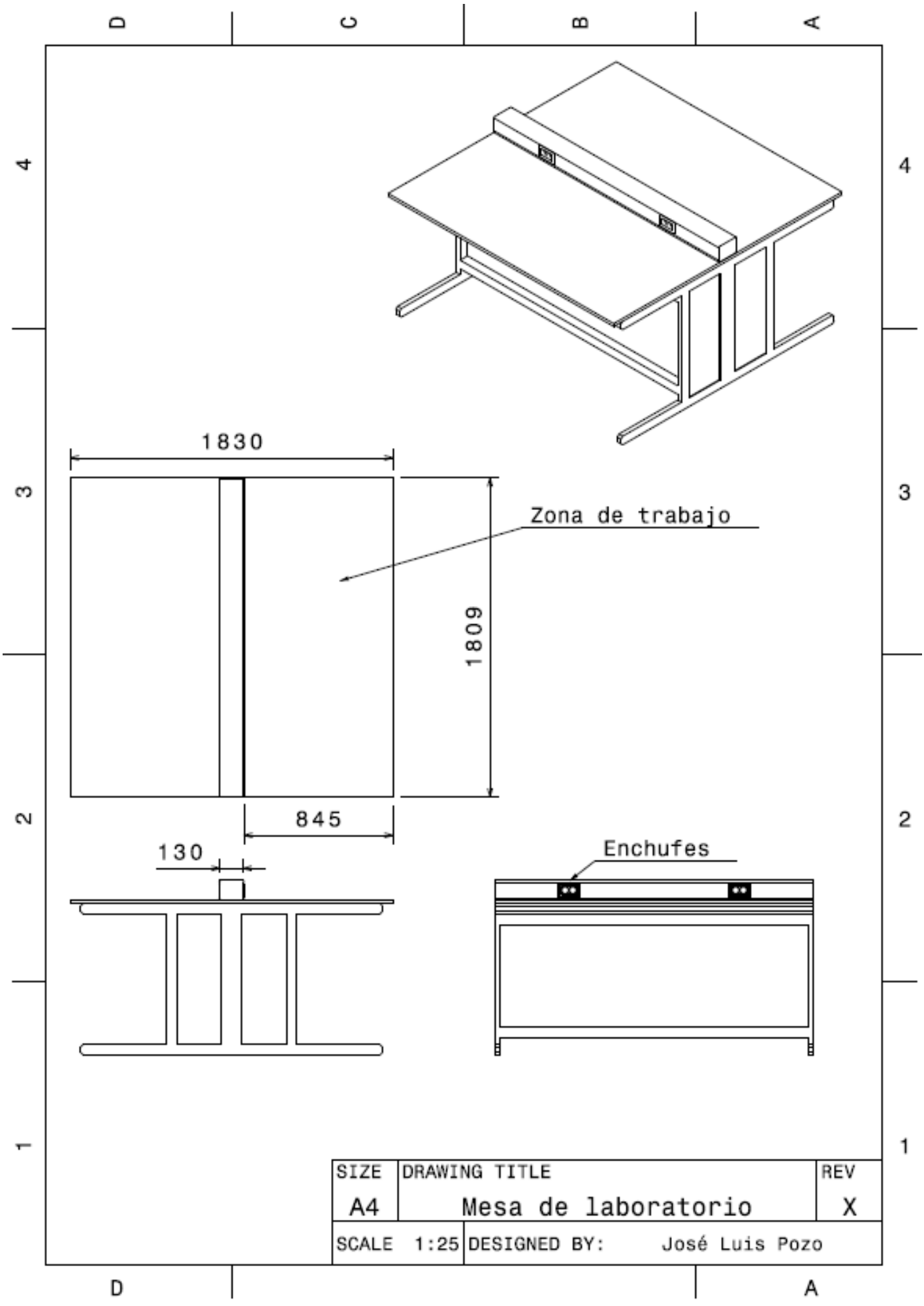


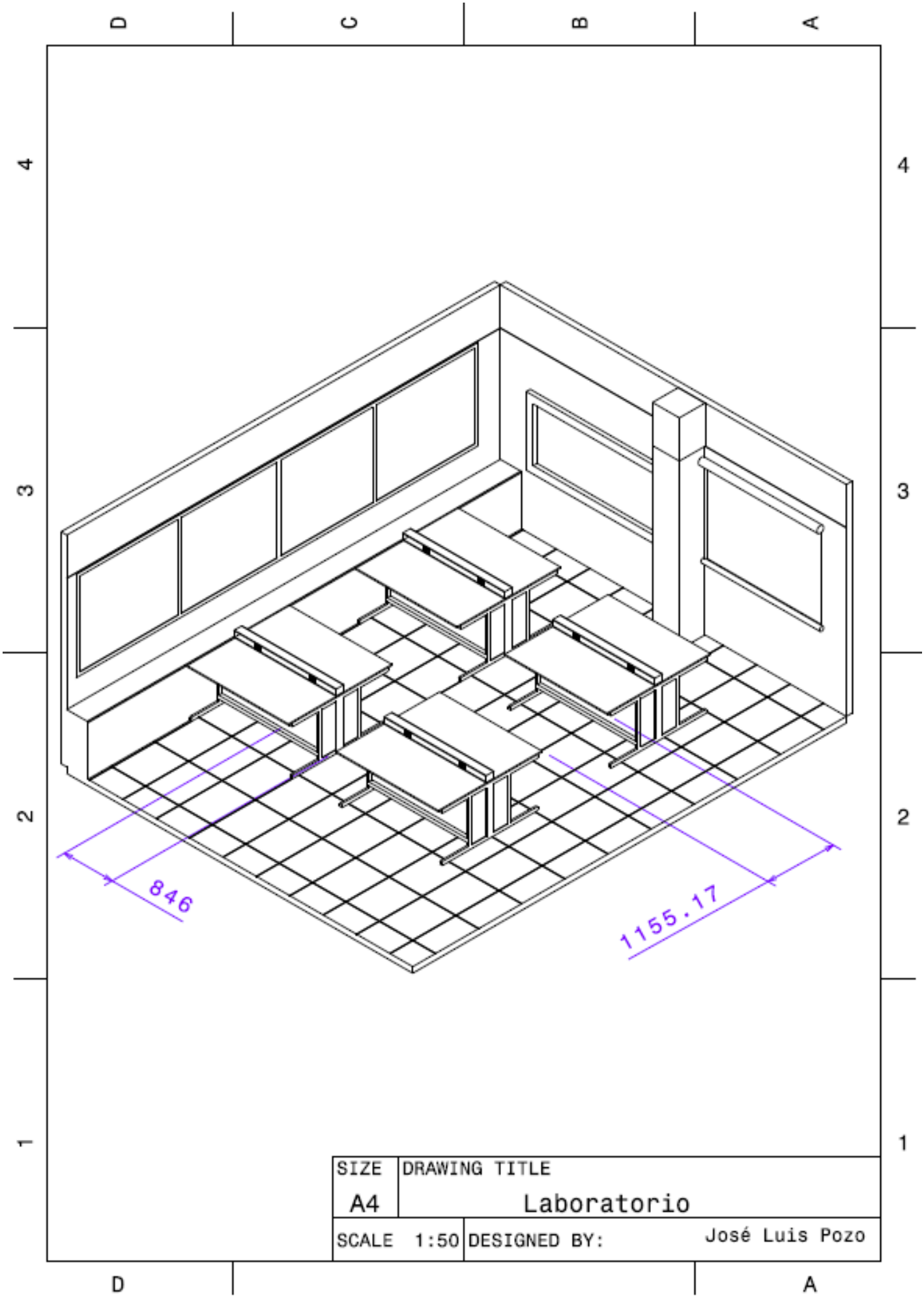




SIZE	DRAWING TITLE
A4	Caja PCB
SCALE 1:2	DESIGNED BY: José Luis Pozo


D
C
B
A
4
3
2
1
44
168
144
1.5
SIZE
A4
DRAWING TITLE
Tapa PCB
SCALE 1:2
DEDIGNED BY: José Luis Pozo


D
C
B
A
4
3
2
1
1830
1809
Zona de trabajo
845
130
Enchufes
SIZE
A4
DRAWING TITLE
Mesa de laboratorio
REV
X
SCALE 1:25
DESIGNED BY: José Luis Pozo


D
C
B
A
4
3
2
1
846
1155.17
SIZE
A4
DRAWING TITLE
Laboratorio
SCALE 1:50
DESIGNED BY:
José Luis Pozo